TOBIAS JESCHKE

BACHELORTHESIS

„REALISATION EINES ONLINE-SHOPS MIT SHOPWARE 4“

SOMMERSEMESTER 2014

BETREUER: PROF. DR. TOM RÜDEBUSCH

Inhaltsverzeichnis

1. Motivation

Im Rahmen meiner langjährigen Tätigkeiten bei der Agentur Equinoxe habe ich umfangreich Erfahrung mit E-Shops gesammelt. Dabei handelte es sich jedoch meist um hochspezialisierte, proprietäre Shops in geschlossenen Systemen. Im Rahmen meines Studiums habe ich meine Kenntnisse in diesem Bereich erweitert.

Motivation

Die eigentliche Motivation dieses Projekts entstand im Rahmen der Vorlesung „E-Business Systeme“ im Sommersemster 2012, bei dem ein Onlineshop mit OXID eShop CE (Version 4.6) umgesetzt wurde, bei dem ich erstmalig Erfahrungen mit einer modernen E-Shop Standard-software mit Ausrichtung auf den öffentlichen Markt gesammelt habe.

Seitens meines Arbeitgebers wurde ich nachfolgend beauftragt, weitere gängige Shopsysteme sowohl allgemein als auch hinsichtlich der Eignung für www.sunrise.de zu prüfen, um eine geeignete Plattform für den Relaunch von www.sunrise-versand.de auszuwählen, mit der ich diesen dann auch umsetzte.

Hinzu kommt, dass das durch diese Arbeit gewonnene - Know-How auch für andere Projekte - ausserhalb Shopware - geeignet ist.
Da Shopware einen sehr hohen technischen Standard hat und mit vielen modernen und leistungsstarken Frameworks arbeitet, kam der Aufgabe der Systemarchitekturanalyse die Rolle des „Learn-from-the-Pros“ zu.

Das bei der Arbeit gewonnene Know-How soll sowohl der Agentur „Equinoxe“ einen Wettbewerbsvorteil bei E-Commerce-Projekten verschaffen, wie auch zu meinem beruflichem Werdegang beitragen.

2. Die Agentur Equinoxe

Das Unternehmen Equinoxe besteht seit 1995.
Es handelt sich um eine inhabergeführte Gesellschaft mit
9 Mitarbeitern und Sitz in Freiburg im Breisgau.

Als Internetagentur und Serviceprovider ist Equinoxe seit 1995 einer der Innovationsführer der Branche.
Equinoxe entwickelt Strategien, Konzepte und Designs
für kundenspezifische innovative Online– und Multimedia-Lösungen.
Ergonomie, Informationsökonomie und der Nutzen für Kunden
und Anwender stehen dabei im Vordergrund.

Agentur Equinoxe

Führende Unternehmen als Kunden.
In den vergangenen Jahren hat Equinoxe erfolgreich Marktführer in den Bereichen Markenartikel und Investitiongüterindustrie als Kunden und Produktmandanten gewinnen können.
Dazu zählen Unternehmen wie Altana, Argus, AXA, General Motors Europe, Siemens AG, Chevrolet, Schuler AG, Hyundai, Helvetia, Opel, KIA, UBS, Messe Hamburg, STIHL, Schindler Aufzüge, Pistenbully, Zehnder AG u.v.m..

Equinoxe besitzt Beratungs-, Design- und betriebssystemübergreifende Programmierkompetenz. Die lange Erfahrung in diesem sich sehr schnell entwickelnden Technologie-Marktsegment und die konsequente Investition in Mitarbeiter-Know-How stellen eine außerordentlichen Wettbewerbsvorteil dar.

Shortfacts

- Unternehmensgründung 1995
- Inhabergeführte Gesellschaft
- 9 Mitarbeiter
- Ausbildungsbetrieb, IHK und BA Mannheim
- redundantes Network Operation Center (1,8 Gbit Glasfaser)

3. Abkürzungsverzeichnis

AGOF

Arbeitsgemeinschaft Online Forschung e.V.

WNK

Zum Weitesten Nutzerkreis (WNK) eines Mediums gehören alle Personen, die innerhalb eines bestimmten Zeitraums bzw. einer bestimmten Anzahl von Ausgaben für dieses Medium eine Nutzungswahrscheinlichkeit haben, die größer als Null ist

Begriffs-klärungen

CRM

„Das Customer RelationshipManagement (CRM) oder Kundenbeziehungsmanagement stellt die Pflege der Kundenbeziehung in den Mittelpunkt. Es geht dabei um das Gestalten und Bewirtschaften der Beziehung zwischen einem Unternehmen oder einer Organisation und ihren Anspruchsgruppen (Stakeholder)“ [14]

4. Begriffsklärungen

Ajax - Einsatz von asynchronen Javascript-Requests zum Laden von (Teil)elementen einer Website oder eines Webservice

Annotation - Das Konzept von Doctrine erlaubt es, via standardisierter Kommentare in PHP Klassen sämtliche Beziehungen zu logisch verknüpften Daten banktabellen zu konfigurieren

API - Programmierschnittstelle. Wird oft gleichbedeutend mit der API-Dokumentation verwendet.

Backend - Der funktionale Teil eines Webanwendung, zu dem in der Regel nur Content-Redakteure Zugang haben

Constraint - streng genommen eine Regel zur Sicherstellung von Datenkonsistenz in Datenbanken (z.B. durch Verwendung von eindeutigen Schlüsseln) - bei Symfony 2 jedoch für das Konzept von Prüffunktionen für beliebige Inhaltstypen verwendet

Cross-Selling - Bei Cross-Selling geht es darum, zusätzlich zu einem Produkt noch weitere zu verkaufen. Sei es mit Produktempfehlungen oder mit passenden Zubehörartikel zum gekauften Produkt.

CSV-Datei - Kommaseparierte Datei, wird für den Import / Export von Daten verwendet.

Doctrine - modernes PHP Framework für die Implementierung von „Model“-Aspekten einer MVC Anwendung. Konkret: Ein objektrelationer Mapper mit zahlreichen Zusatzfunktionen. Dieser erlaubt eine zentrale Definition der Datenbankstruktur in PHP. Zusätzlich bietet Doctrine die Möglichkeit, alle Queries in einem System zentral in den songenannten Repositories zu definieren, um diese an verschiedensten Stellen im System wieder verwenden zu können.

Entity - .Eine Entität kann man mit der Instanz bei der objektorientierten Program mierung vergleichen. Bei Doctrine ist eine Entity eine Einheit aus Klassen instanz und zugehörigem Datenbank-Datensatz.

Event - „Definierte Ereignisse, die im Workflow des Shops auftreten bzw. auftreten können. An jedem Event kann man so genannte Event-Listener registrieren - dies sind eigene Funktionen, die automatisch benachrichtigt werden, sobald das jeweilige Event ausgelöst wird. (Bsp.: Abschließen einer Bestel lung“)

Ext JS - Von der Firma Sencha entwickelt, ist Ext Js eines der grossen Frameworks für funktionale Weboberflächen. Es kann als klassisches View-Framework für MVC Architekturen angesehen werden.

Frontend - „vorderer Teil“ einer Webanwendung, also derjenige, der öffentlich im Internet zu sehen ist

Hook - Überschreiben / Ergänzen von Funktionalitäten eines Systems

Konversion - Im Kontext „E-Commerce“: Abbruch des Shoppings

Mapping - Mittel, um festzulegen, was wohin gehört.

Model - Der Teil einer MVC Architektur, der für die Datenhaltung und die Geschäfts logik zuständig ist. Bei Doctrine / Shopware : konkretes Element, welches instantiiert werden kann.

MVC - klassisches Entwurfsmuster beim Software Engineering, welches eine klare Arbeitsteilung zwischen drei funktionalen Parteien (Model, View, Controller) vorsieht

Namespace - Konzept von PHP 5.x, bei dem Funktionen und Variablen ein sprechen der Gültigkeitsraum zugewiesen wird

ONYX - universelles, international benutztes Metadatenformat für Bücher wird meist mittels XML umgesetzt

ORM - Produkt von Doctrine, jedoch wird „Doctrine“ of gleichbedeutend mit Doctrine‘s ORM verwendet

Package - Paket einer Codebibliothek

Persistenz - Dauerhaftigkeit einer Datenoperation. Wichtiger Begriff bei Doctrine.

Proxie - bildlich: ein einem Element vorgeschaltetes Element, dass nach außen hin dessen Funktionen übernimmt und erweitert. Bei Shopware: durch Hooks zur Laufzeit manipulierte Klasseninstanzen**e**

Query - Datenbankabfrage

Up-Selling - Ist der Versuch, einem Käufer beim Kauf ein höherwertigeres Produkt zu verkaufen

5. Marktanalyse

5.1 - Bestandsaufnahme E-Commerce in Deutschland

Der seit einigen Jahren existierende wirtschaftliche Aufwärtstrend von E-Commerce hält weiterhin an – die Entwicklung im Jahr 2013 wurde dabei als besonders günstig eingeschätzt.

Marktanalyse

Die Grafiken „E-Commerce-Umsatz in Deutschland seit 19999“ [7] sowie „Umsätze mit Büchern im Versandhandel“ [30] zeigen deutlich, wie es mit der Entwicklung in Deutschland in den vergangenen Jahren bestellt ist - sowohl im Bereich E-Commerce allgemein, wie auch im Buch-Versandhandel.

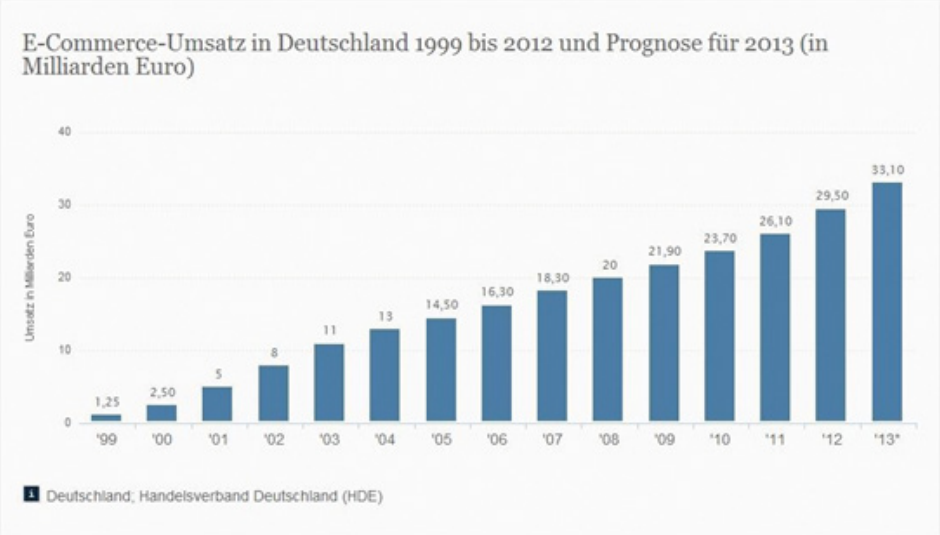


Abbildung „E-Commerce (DE) - Umsatztrend seit 1999“ [7]

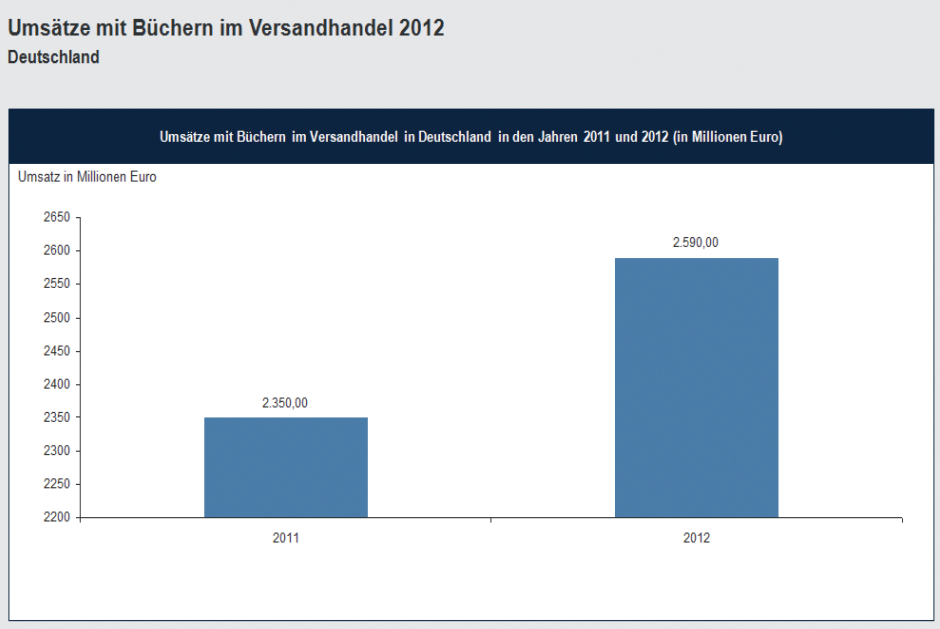


Abbildung „Deutschland - Umsätze mit Buechern im Versandhandel [30]

Für den konkreten Fall „Sunrise“ kann daraus abgeleitet werden, dass ein Relaunch des Online-Shops vielversprechend ist, da E-Commerce in Deutschland nach wie vor ein Wachstumsmarkt ist.

Die weltweite Entwicklungs spiegelt diese Beobachtung wieder - auch wenn dies - für den überwiegend auf den deutschen Markt ausgerichteten Shop www.sunrise-versand.de nicht unmittelbar eine Rolle spielt. Ausserdem bedeutet dies - aus Perspektive der ausführenden Agentur „Equinoxe“ - dass Know-How im Bereich „E-Commerce“ weiterhin von grosser Bedeutung im Agenturalltag sein wird.

5.2 - Online-Potentiale des deutscher Buchhandels

Grundsätzlich ist die allgemeine Lage des deutschen Buchhandels besser, als man es vielleicht erwarten würde.
Laut einer Marktstudie der AGOF aus dem Jahre 2010 haben 58,6 Prozent der Internetnutzer im deutschen Raum ein Interesse an Büchern [1], das entspricht 29,10 Millionen Nutzern.
Im gleichen Zeitraum haben 41,2 Prozent der deutschen Internetnutzer Bücher im Internet gekauft, das entspricht 20,47 Millionen Nutzern.

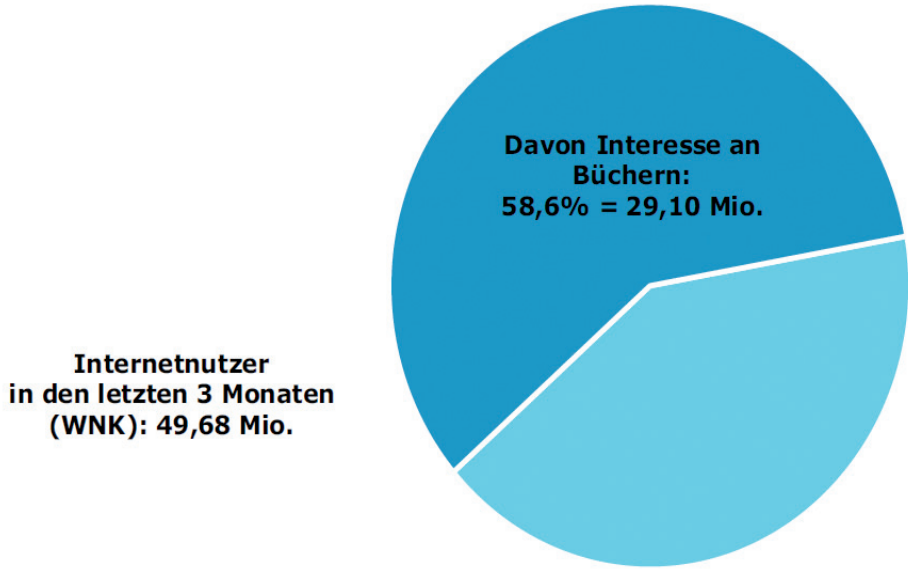


Abbildung „Anteil Interesse Bücher an der Gesamtheit der deutschen Internetnutzer 2010“ [13]

Untersucht man den gleichen Sachverhalt 2013 erneut [49] , so ergibt sich sogar eine Steigerung des Interesses an Büchern um 7,15 % auf 30,91 Mio. Nutzer.

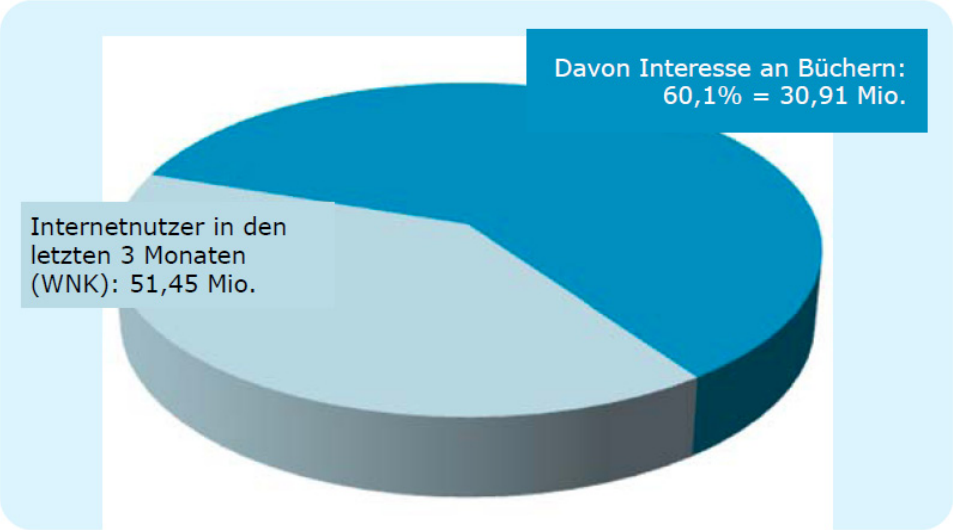


Abbildung „Anteil Interesse Bücher an der Gesamtheit der deutschen Internetnutzer 2013“ [14]

Auch bei den tatsächlichen Käufen ist der Trend für die Buchbranche positiv: Gemäß einer Auswertung im E-Commerce Leitfaden 2012 (basierend auf Daten der AGOF) werden im Internet überwiegend physische Produkte gekauft, wobei Bücher mit 43,4 % die Spitzenreiterposition einnehmen [16].

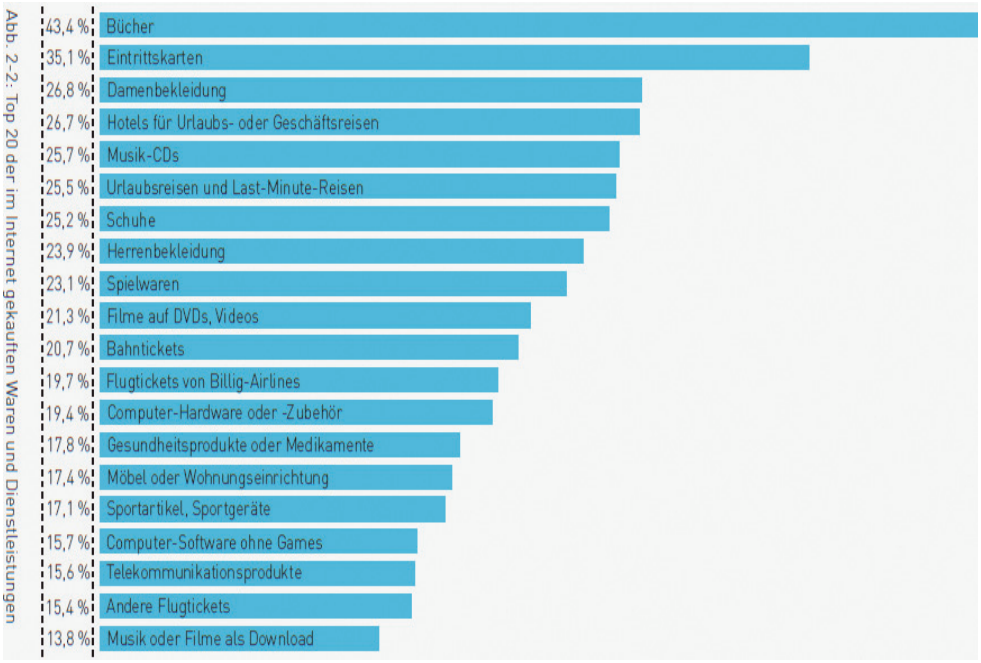


Abbildung „Liste der Top 20 E-Commerce Produkte in Deutschland 2012“ [1]

Im Klartext: Die Lage für den Vertrieb von Büchern im Internet kann als „gut“ bezeichnet werden.

5.2 Angstgegner Amazon – wie kann sich ein kleiner Fachhandel wie Sunrise den Versandriesen stellen ?

Die rosigen Zahlen für den Online-Buchhandel sollen jedoch nicht darüber hinwegtäuschen, dass ein Großteil der Umsätze von Anbietern wie Weltbild, Hugendubel und Amazon gemacht werden.
Auch wenn keine genauen Zahlen über die Marktverteilung vorliegen, geht der Verband der Versandhändler BVH allein für Amazon von Zahlen von um die 20% Marktanteil für den Zeitraum 2012 aus.

Ausgangs-situation

Daher stellt sich die Kernfrage:
Wie kann sich ein familienbetriebenes Kleinunternehmen wie Sunrise angesichts dieser Konkurrenz am Markt behaupten ?

6. Sunrise - Der Kunde

6.1. Ausgangssituation

Der Kunde - Charakterisierung, Besonderheiten und Einordnung

Auftraggeber des Projekts ist das Berliner Fachgeschäft „Sunrise“, welches Homöopatische Literatur sowie Zubehör für die homöopatische Praxis vertreibt.
Dies geschieht sowohl über ein Ladengeschäft in Berlin wie auch den Online-Shop www.sunrise.de.
Das Unternehmen wird von dem Ehepaar Dr. Bernd und Angelika Henne, beides Spezialisten im Bereich Homöopathie, betrieben.
Hr. Henne rundet das Angebot durch mobile Büchertische ab, mit denen er die Republik an den Wochenenden bereist.

Fassen wir zunächst die Stärken von Sunrise zusammen:

- Sunrise wird von zwei Menschen betrieben, die über ein hohes Maß an Spezialwissen verfügen – die Homöopathie ist Ihre Passion (Dr. Bernd Henne ist Chemiker, Heilpraktiker und Homöopath, Frau Henne ist Industriekauffrau und autodidaktische Homöopathin)

- Der Artikelkatalog von Sunrise besteht aus „handverlesenen“ Büchern und Zubehörartikeln für die homöopathische Praxis, was eine hohe Authentizität gewährleistet

- Dadurch hebt sich Sunrise von den großen Versandhäusern positiv ab, die zwar ein riesiges Sortiment führen, jedoch über kein hauseigenes Know-How verfügen – dieses wird meist über die Nutzer-Produktbewertungen abgewickelt.

- Herr Henne bereist mit seinen homöopathischen Büchertischen die Republik, ein zusätzliches Medium zur Vertrauensbildung

Dem gegenüber stehen die Schwächen von Sunrise:

- Nur wenige Mitarbeiter, die gleichzeitig das Ladengeschäft betreiben
- Geringes Kapitalvolumen

Ausgangs-situation

6.2 Die Zielgruppe

Die Zielgruppe besteht laut Aussage von Sunrise in erster Linie aus den langjährigen Stammkunden, die fast alle einen medizinischen Hintergrund (Ärzte, klassische Homöopathen, Heilpraktiker, Hebammen) haben oder sich privat mit Homöopathie befassen.

Das durchschnittliche Alter liegt um die 40, oft darüber. Frauen bilden bei der Kundschaft die Mehrheit [1].
Eine Zielgruppe, die stark technisch versiert ist oder großen Wert auf ein hippes Design legt, ist nicht gegeben. Vielmehr gilt es, eine Lösung für eine Zielgruppe zu erarbeiten,welche möglichst klar im Design und selbsterklärend in der Bedienung ist.

Die Gefahr, Kunden durch eine zu große Umstellung zu verlieren, ist genauso vorhanden wie das Potential, mit einem neuen, „frischen“ Shop neue Kunden zu gewinnen.

6.3 Stärken ausreizen - Konzentration auf das Kerngeschäft

Sunrise bietet als Fachhändler einen entscheidenden Vorteil gegenüber Shops, die ihr Produktportfolio in alle Richtungen erweitern und dabei zum einen den Look-and-Feel eines Gemischtwarenhändlers erlangen und sich dabei meist um die Gelegenheit bringen, sich mit den verkauften Artikel wirklich auszukennen.
Durch die klare Zielgruppe und das fachliche Know-How, über welches Sunrise verfügt, kann die Ausrichtung auf eine Stammkundschaft aktiv vorangetrieben werden. Dazu zählen seitens Sunrise erstellte Kommentare zu einem Produkt wie auch Informationen über Neuerscheinungen oder besonders empfehlenswerte Artikel.

6.3 The „bad old times“ - der bestehende Shop

Der bestehende Shop war eine Insellösung aus dem Jahr 2004, die für den damaligen Firmenbesitzer Dr. Wolfgang Schmelzer seitens Equinoxe realisiert wurde. Dabei ging es vornehmlich darum, eine bereits damals veraltete Filemaker-Pro Datenbank „shopfähig“ zu machen und war mit einem proprietärem Cold-Fusion-CMS realisiert.
Das Layout und die gesamte Funktionsweise war veraltet und konnte nur noch durch einen Relaunch ersetzt werden. In Zeiten, bei denen Online-Shops mit Amazon und Co mithalten müssen, um am Markt zu bestehen, werden vom Kunden nur noch moderne Shop mit guter Usability und komfortablem Bestell- und Bezahlmanagement angenommen.

Ausgangs-situation

Eine weitere Herausforderung war, den bestehenden Artikelbestand ins neue System zu überführen, und die alte, verschachtelte Kategoriestruktur in eine neue, flache Kategoriestruktur mit kombinierten Produktfiltern zu überführen.

6.4. Chancen des Projekts aus Kundensicht

Für die Shopbetreiber war der Wechsel auf ein anderes Shopsystem unumgänglich, um dem Konkurrenzdruck seitens Amazon & Co etwas entgegenzusetzen. Gleichzeitig bot sich die Möglichkeit, beim Relaunch alte und verkrustete Strukturen auszumerzen und durch neue und bessere auszutauschen. Gleichzeitig nähert man sich dem Kunden an, der nach dem Relaunch allen Komfort und Intelligenz vorfinden soll, den ein zeitgemäßes Shopsystem bietet.

Die Erweiterbarkeit in Richtung ERP-Anbindung, möglicherweise sogar in Richtung automatisierte Katalogerstellung, birgt wichtige Möglichkeiten für die zukünftige Entwicklung von Sunrise.
Auch die Option, das Produktsortiment mit E-Books aufzustocken, ist für einen Buchhändler der Neuzeit ein wertvoller Schritt in Richtung Zukunftssicherheit.

6.5. Zieldefinition aus Kundensicht

Eine zentrale Frage beim Relaunch von Sunrise war aus Kundensich demnach: wie kann ein kleines Unternehmen mit begrenzten Kapital- und Mitarbeiterresourcen den Sprung zu einem modernen, hochwertigen Shop mit ähnlichen Features (intelligente Produktsuche, Produktfilter, Bewertungssystem etc.) wie die großen Konkurrenten am Markt schaffen?

6.6. Anforderungsprofil des Kunden

Zu Projektbeginn war es – abgesehen von den gerade genannten Faktoren – für Sunrise sehr wichtig, eine Shopsoftware auszuwählen, welche Schnittstellen zu anderen Systemen - wie beispielweise einem E-Book-Anbieter – bereithält.
Mit dazu gehört die Fähigkeit des Shopsystems, digitale Produkte, also Download-Artikel zu verarbeiten.

Anforderungsprofil „Kunde“

Grundsätzlich sollte der Shop von der Usability und der sog. „User Experience“, also dem Bedien-und Kauferlebnis her herausstechen - was nicht ausschliesst, dass klassische Muster verwendet werden -und vom „Look and Feel“ auf der Höhe der Zeit sein.
Klassische moderne Produktangebote wie Cross- und Up-Selling, Produktmerkzettel, Produktbewertungen sollten bereitgestellt werden.
Die Wichtigkeit einer intelligenten Suche wurde vom Kunden als hoch eingestuft.

Auch soll langfristig die Möglichkeit bestehen, den Shop wahlweise mit zugekauften Modulen oder Individualprogrammierung zu erweitern, was für einen Open Source Shop spricht.

Von zentralem Interesse war jedoch eine durchdachtes, selbsterklärendes Bediensystem, da die Shopbetreiber Reibungsverluste bei der Umstellung nach Möglichkeit vermeiden wollten.
Dieser Anspruch galt sowohl für das Backend als auch das Frontend.
Meine Aufgabe bestand also unter anderem darin, ein Shopsystem auszuwählen, welches „out-of-the-box“ den Vorstellungen des Kunden möglichst stark entsprach.

Die zugehörige Dokumentation wie auch die Community sollte möglichst stark sein, damit sowohl agenturintern als auch später beim Kunden selber eine möglichst hochwertige und lückenlose Knowledgebase zur Verfügung steht.
Hinzu kommen Anforderungen, die sich aus den Erfordernissen des bestehenden Shops ableiten:

- Unterstützung von >= 10000 Artikeln
- technisch aktueller Stand für schnellen und stabilen Shop
- Übersichtlichkeit, sowie im Backend wie auch im Frontend-Template
- Übersichtliches CMS
- Intuitive Bedienung
- Möglichkeiten für Suchmaschinenoptimierung

7. Equinoxe - Perspektive „Agentur“

7.1. Chancen und Risiken des Projekts aus Agentursicht

Aus Perspektive der Agentur Equinoxe bietet das Projekt sowohl einen aktuellen E-Commerce-Softwareüberblick wie auch einen handfesten Einstieg in den Umgang mit Shopware 4.
Hierbei ging es nicht nur um die konkrete Projektumsetzung als reinen Kundenauftrag, sondern um die Sondierung und Evaluierung einer Shopsoftware, die ausreichend skalierbar für den Bedarf bei weiteren E-Commerceprojekten ist.
Auch ist der Wissenszuwachs bei den beteiligten PHP Frameworks projektübergreifend von Nutzen. Der Mehraufwand für die initiale Einarbeitung beim Projekt „Sunrise“ wurde bewusst vor dem Hintergrund dieser Kriterien in Kauf genommen.

Anforderungsprofil „Agentur“

7.2. Anforderungsprofil Equinoxe

Für eine Agentur wie Equinoxe, die Projekte bis ca. 2010 noch vorwiegend mittels selbstentwickelter Produkte umsetzte, dann jedoch strategisch den Fokus auf Open-Source legte, ist ein fundiertes Know-How leistungsstarker und flexibler Open-Source-Anwendungen von großer Bedeutung.
Im Bereich E-Commerce galt es, eine moderne, vielversprechende E-Shop-Software zu evaluieren, um die Eignung für den weiteren Einsatz in der Agentur zu prüfen.
Das Projekt Sunrise hatte dabei Pilotcharakter – es sollte nicht nur das Projekt im Kundenauftrag umgesetzt werden, sondern auch eine moderne E-Shop Software möglichst erfolgreich in der Praxis getestet werden, um diese nachfolgend ins Angebotsportfolio der Agentur aufzunehmen.

Seitens Equinoxe bestanden folgende Anforderungen an eine E-Shop Software:

- **Open Source als Plattform**
  - Einsicht in den Code zum besseren Verständnis
  - Einsatz auch bei kleineren Projekten ohne Lizenzgebühren möglich
  - Integration von manuellen Anpassungen möglich

- **Technik**
  - möglichst MVC basierend
  - Verwendung von Symfony 2 (mächtiges PHP Framework mit immer größerer Verbreitung - Drupal 8 stellt einen Großteil seiner Kernarchitektur darauf um)

- **Programmiersprache**
  - PHP 5.x – gängigste und leistungsstarke Plattform im Open Source Bereich
  - entsprechendes Know-How betriebsintern verfügbar

- **Datenbank**
  - Unterstützung der Open Source Datenbank mySQL

Anforderungsprofil „Agentur“

- **Standardmäßige, professionelle Implementierung von unverzichtbaren Basiskomponenten wie:**

  - intelligente Produktsuche
  - Artikelverwaltung mit hoher Usability / Mehrfachkategoriezuordnung
  - Produkteigenschaften / Produktfiltern
  - Aussagekräftige und leicht zu bedienende Auswertungswerkzeuge
  - Medienverwaltung
  - moderne Schnittstellen, bevorzugt REST-basierend
  - hohe Usability beim Bestellprozess
  - hohe Qualität des Standardtemplates
  - Empfehlungssysteme (Up- und Cross-Selling, Produktbewertungen)

8. E-Shop Systeme

8.1. Definition - was leistet ein E-Shop System ?

Ein E-Shop System leistet dieses: „E-Shops bieten eine Möglichkeit, die Anbahnung von Transaktionen (aus den Teilbereichen Information, Vereinbarung, Abwicklung und Service bestehend) zu initiieren und zu unterstützen bzw. gänzlich elektronisch abzuwickeln. Dabei wird eine Plattform geschaffen, auf der Anbieter ihre Waren oder Dienstleistungen präsentieren und der Interessent die Handhabe besitzt, Produktinformationen einzuholen.“ [50]

Dabei spielt es durchaus eine Rolle, wie dies technisch umgesetzt ist, wieviel für das System finanziell aufgewendet werden muss, und wie hoch der Einarbeitungsaufwand ist. Al-Qirim, Nabeel A. Y. formuliert dies in seinem Buch „Electronic commerce in small to medium-sized enterprises“ [3] so: „The technologies and applications must be easy and relatively cheap. [...] While some IT learning is necessary and beneficial for small business, a significant investment of time is not attractive“.

8.2. Evaluierungskriterien - wie können E-Shop Systeme grundsätzlich eingestuft werden ?

8.2. 1 Anforderungen und Vorauswahl

Nach einer Definition von Thayer und Dorfman sind Anforderungen „vom Anwender benötigte Fähigkeit eines Systems, um ein Problem zu lösen oder ein Ziel zu erreichen […]“. [15]
Keyvans Karimis Ergänzung, dass bei E-Shop-Anwendungen zwei Anwendertypen, den Endkunde sowie den Shopbetreiber berücksichtigt werden müssen, ist jedoch für E-Shop Software zutreffend.[15]
Im vorliegenden Fall galt es daher , die Bedürfnisse des Shopbetreibers, dessen Kunden wie auch der Partner-Agentur Equinoxe zu erfüllen.

E-Commerce Systeme

Evaluierungs-kriterien

8.2.2 Evaluierungskriterien

Für die konkrete Auswahl der verwendeten E-Shop-Software wurden mögliche Standard-Software nach relevanten Gesichtspunkten geprüft. Diese sind:

Zukunftssicherheit

Da – gerade für eine kleines Unternehmen wie Sunrise – Folgekosten kalkulierbar sein müssen, ist die Zukunftssicherheit einer Shop-Software ein entscheidender Faktor bei der Auswahl.
Das fängt bei der simplen Überlegung an, ob es die Software in 3-5 Jahren noch geben wird und geht weiter bei der Einschätzung, welchen Funktionszuwachs der Shop in einem Jahr erfahren wird.

Gleiches gilt für die Aktualität der Software – wird diese nur in großen zeitlichen Abständen weiterentwickelt, droht die Gefahr, dass das System nicht mehr zeitgemäß ist und erneut ersetzt werden muss – mit dem einhergehenden zeitlichen und finanziellen Aufwand für den Betreiber.
Dies kann auch dann kritisch sein, wenn Plugins und Schnittstellen zu für den Geschäftsbetrieb der IT-Gesamtlandschaft wie ERP Software, Zahlungsschnittstellen nicht mehr verfügbar sind oder mit einigem Aufwand angepasst werden müssen.

Umgekehrt bedeutet dies, dass die Wahrscheinlichkeit, mit der für ein Shopsystem Plugins für die Kommunikation mit anderen Systemen existieren, mit der Akzeptanz und Verbreitung des E-Shopsystems signifikant steigen.

Integration in die bestehende IT-Landschaft

Grundsätzlich ist die Eignung einer Shopsoftware für die Integration in bestehende technische Strukturen ein wichtiger Faktor, da beim vorliegenden Projekt jedoch ein kompletter Neubeginn angestrebt wurde, konnte dieser weitestgehend außen vor bleiben.
Ich habe vornehmlich geprüft, ob es Plugins für die Anbindung gängiger und leistungsstarker ERP-Systeme gibt, damit eine Anbindung eines modernen ERP – wahlweise innerhalb des Projekts oder zu einem späteren Zeitpunkt – keine Schwierigkeiten darstellt (vgl. Kapitel „Der Idealfall - integriertes ERP“).

E-Commerce Systeme

Evaluierungs-kriterien

Schnittstellen

In der Vorlesung „E-Business Anwendungen“ wurde die Wichtigkeit von Schnittstellen im allgemeinen gelehrt. Denn – kaum ein Shop kann als reine Insellösung funktionieren - meist werden mindestens Schnitstellen zu Zahlungsanbietern benötigt - weitere Schnittstellen zu einem ERP-System oder für den Datenimport / Export kommen hinzu.

Ein E-Shop, der aus Endkundensicht optimal funktioniert, jedoch aufgrund von fehlenden Schnitstellen für den E-Shop Betreiber hohe Kosten für die Abwicklungen von Transaktionen verursacht, wäre nicht als erfolgreich zu betrachten. [15]

Bei der Evaluierung von Schnittstellen sind diejenigen zu bevorzugen,die standardisiert sind und dementsprechend gut dokumentiert sind [16]. Julien Ardisson, Produktmanager bei Strato, geht bei der Beratung von E-Commerce Projekten noch einen Schritt weiter: „Schnittstellen sind am wichtigsten, vor allem zu verschiedenen Bezahlmöglichkeiten.“ [16]

Bei der Evaluierung von E-Shop Software habe ich zwei Arten von Schnittstellen untersucht - zum einen Schnittstellen zur Zahlungsabwicklung (wobei eine möglichst grosse Anzahl an unterstützter Anbieter als positiv eingestuft wurden), zum anderen Schnittstellen zum Content-Transfer, wie beispielsweise für E-Book Datentransfer benötigt wird. Gängige Techniken hierfür sind sogenannte Webservices, die als Schnittstelle zwischen Systemen dienen.

Im Wesentlichen haben sich zwei Webservice-Ausprägungen herauskristallisiert:

- SOAP - ein Protokoll für den Austausch von Nachrichten, wobei jede Nachricht als XML definiert ist
- REST - eine HTTP-basierende Schnittstelle, bei der Resourcen (= Daten) über eine URI angesteuert werden kann

Open Source als Methodik •

Die Kernidee von Open Source ist diese:
„Konzept, nach dem Programme mit ihrem Quellcode ausgeliefert werden. Jeder darf den Quellcode einsehen und verändern. Die Open Source Initiative (OSI) definiert Kriterien, die Open Source Software erfüllen soll“ [11].
Dabei werden nicht nur Methodikansätze wie den freien Vertrieb, den offenen Quellcode, die Möglichkeit zur Veränderung des Codes wie die Ableitung weiterer Anwendungen definiert, sondern auch ethische Grundsätze wie Abgrenzung gegen Diskriminierung definiert – Ansätze, die man bei kommerziellen Softwarekonzernen meist vergeblich sucht.

Die Bedeutung von Open Source hat in den letzten 10 Jahren enorm zugenommen. War der Begriff Open Source in den Jahren vor 2008 noch mit Assoziationen wie „Bastler-Applikationen“ oder „Nischenprodukt“ behaftet, so ist dieser vor dem Hintergrund von Erfolgsgeschichten wie mySQL oder der Veröffentlichung des OXID E-Shops als Open Source Version 2008 zu einem ernstzunehmenden Methodikansatz geworden.

Bereits das heute zentrale HTTP wurde (angestoßen von Tim Berners-Lee) in einem offenen Prozess entwickelt, zu dem viele Entwickler weltweit beitrugen. Im Open Source Jahrbuch 2008 findet sich zum Thema mySQL das folgende Statement:

„Die wertvollste Art der Teilnahme galt der Qualitätsverbesserung der Software. Die Anwendergemeinde setzte die Datenbank schon früh unter Bedingungen ein, die bis dato von den Entwicklern nicht getestet worden waren oder nicht getestet werden konnten.[...] Durch diese Teilnahme unserer Community konnte das Produkt qualitativ wesentlich verbessert werden“[1]. Tim O’Reillys Begriff der „Architecture of participation“ [44] kann daher nicht nur als Mitwirkung der Nutzer bei der Generierung von Web-Inhalten verstanden werden, sondern auch beim Entstehungsprozess von Software.
Statistiken zeigen, dass bereits im Jahr 2008 sowohl die Wichtigkeit (und damit die Akzeptanz) wie auch die Zufriedenheit mit Open Source Anwendungen im deutschen Markt groß war – Tendenz steigend.

Die Vorteile von Open Source sind [5]:

- hohe Flexibilität
  Anpassbarkeit und Erweiterbarkeit,
  die bei proprietärer Software teilweise nicht möglich ist.

- höhere Sicherheit
  Experten aus der Entwicklergemeinde analysieren potenzielle Sicherheitslücken im Quellcode. Treten einmal Lücken auf, werden diese sofort geschlossen.

- höhere Produktqualität
  Eine weltweite Entwicklergemeinde verbessert die Software permanent, ohne Marktzwänge, wie beispielsweise feste Releasetermine.

E-Commerce Systeme

Evaluierungs-kriterien

- keine Lizenzkosten
  Trotz des höheren Anteils der Dienstleistungs- und Hardwarekosten ein spürbarer Vorteil.

- keine Abhängigkeit
  Da die Software frei verfügbar ist, besteht keine Abhängigkeit vom Know-how eines bestimmten Herstellers.

- offene Standards
  Hohe Interoperabilität und Kompatibilität durch offene Schnittstellen.

- Wiederverwendbarkeit
  Hohe Wiederverwendbarkeit von Programmcode und Know-how-Transfer durch Quellcodeanalyse.

Die Agentur Equinoxe setzt schwerpunktmässig auf Open Source bei der Auswahl der Werkzeuge im Projektgeschäft.
Daher wird Open Source in die Reihe der Evaluierungskriterium aufgenommen.

Dokumentation / Community

Da beim Projekt Sunrise - unabhängig von der Auswahl der E-Shop Software - von einer Realisation mittels einer kostenfreien Community Edition ausgegangen wurde, spielt die Dokumentation wie der Community eine wichtige Rolle. Die zentrale Frage hierbei lautet:
wie gut ist der kostenfreie Support ?

Supportmöglichkeiten sind beispielsweise ein Wiki oder gut strukturiertes Fachforen. Gerade bei einer Open-Source-Shopsoftware ist die Lebendigkeit der zugehörigen Community von großer Bedeutung. Ohne diese kann z.B.mit einer Community Version kaum gearbeitet werden, da man ohne Support vom Hersteller auskommen muss.

Mit aus diesem Grund kamen bei der Vorauswahl nur Software-Kandidaten in Frage, die wahlweise als Big Player / Klassiker eingestuft werden können, oder als vitale Newcomer wie Shopware, denn kleinen Nischenanbieter haben meist keine ausreichend aktive Community, die einen Herstellersupport ersetzen kann.

Einschätzung des Softwareherstellers

E-Commerce Systeme

Evaluierungs-kriterien

„Die Einbeziehung des E-Shop Softwareherstellers stellt aufgrund der z.T. hohen Abhängigkeit je nach Betreibermodell, ein wichtiges Kriterium dar.“ [15] Daraus resultiert die berechtigte Forderung, den E-Shop Softwarehersteller allein schon aufgrund der Abhängigkeit von ihm (Update-Politik, Feature-Zuwachs, Dokumentation) eine grosse Rolle zukommen zu lassen. Softwarehersteller, die ihr Produkt dynamisch weiterentwickeln und in regelmässigen Abständen - idealerweise mehrmals pro Kalenderjahr - Major releases veröffentlichen, werden bevorzugt behandelt.

Systemarchitektur

„Die Architektur beschreibt die Struktur eines Systems, dessen Bausteine, Schnittstellen und deren Zusammenspiel. Die Architektur besteht aus Plänen und enthält i.d.R. Hinweise und Vorschriften, nach denen das System zusammengebaut werden sollte. Damit kann die Software-Architektur auch als Plan eines Systems gesehen werden.“[15]

Standardtemplate

Um die Entwicklungskosten möglichst niedrig zu halten, wurden bei der Auswahl Shops bevorzugt, bei denen die Standardtemplates der E-Shop Software den Vorstellungen des Kunden möglichst nah kamen, so dass das Projekt mit Änderungen an einem der Standardtemplates bewerkstelligt werden konnte. Als wichtig wurde definiert, dass das Standardtemplate klar strukturiert und aufgeräumt ist und mit hoher Usability versehen ist.

Usability

Die Wichtigkeit von Usability darf - gerade bei einer E-Shop-Software, bei der es stark darauf ankommt, einen Nutzer im Online-Shop zu „halten“ und Abbruchraten gering zu halten - nicht unterschätzt werden. Einem Essay von Florian Schneider, Creative Director von Netz98 folgend, sind 10 der wichtigsten Ansätze für eine gute E-Commerce-Usability diese [10]:

- **Klassischer, gelernter Grundaufbau**
  Diese Regel wird als wichtigste eingestuft, da mittels ihrer geprüft werden kann, ob ein Shop anlehnend an Interaktionsgrundmuster gestaltet ist, die für den Nutzer essentiell für die grundlegende Orientierung ist.Dazu zählen Dinge wie eine prominent platzierte Suchfunktion oder die Platzierung des Warenkorbs und der Links zu „mein Konto“, „Ausloggen“ rechts oben.

- **Intuitive und visuelle Nutzerführung**
  Da das Nutzungsverhalten in den letzten Jahren deutlich visueller geworden ist, möchte der Nutzer mit klar erkennbaren, intuitiven Elementen geführt werden

- **Inhalte begrenzen, Übersicht bieten**
  Da das menschliche Gehirn bei der Fähigkeit begrenzt ist, Objekte in grosser Anzahl wahrzunehmen, gilt es, überladene Shopseiten zu vermeiden, die zu Shop-Abbrüchen führen können.
  Vielmehr lautet die Maxime bei der Inhaltsgestaltung, wenige, möglichst handverlesene auf der Landingpage bereitzuhalten und bei der eigentlichen Artikelsuche Produktfilter und eine intuitive Suchfunktion zu verwenden. Im Idealfall bewegt sich der Nutzer fliessend durch das Angebot, ohne dies überhaupt zu bemerken.

- **Stringenz**
  Der Begriff „Stringenz“ definiert das Bedürfnis einen Menschen, gelernte Interaktionsmuster auch anwenden zu dürfen.
  Dazu zählen Dinge wie einheitliche Farbgebung von Buttons oder gleiche Platzierung von prominenten Interaktionselementen wie dem Warenkorbbutton.

- **Priorisierung der Elemente**
  Ist beim klassischen Screen-Design ein homogenes Design wünschenswert, so ist es beim E-Commerce unabdingbar, Interaktionselementen eine Wertigkeit zu verleihen.
  Sprich: Elemente, die zum eigentlichen Produktauffindungs- und Kaufprozess gehören, müssen aus der Gesamtmenge der Interaktionselemente hervorstechen.

- **Unterschiedliche Einstiege bieten**
  Je nach Nutzer-Vorlieben und Eigenart wird beim Shopping die klassische Navigation, die Suchfunktion oder ein grafisches Leitsystem verwendet. Hinzu kommen Elemente, die heutzutage ebenfalls vom Nutzer als „gelernt“ vorausgesetzt werden können, wie beispielsweise Up- oder Crossselling.
  Für die Evaluierung bedeutete dies, dass E-Shop Systeme bevorzugt wurden, die möglichst viele Einstiegsmöglichkeiten bieten.

- **Orientierung**
  Für den Nutzer soll jederzeit erkennbar sein, wo er sich innerhalb des Use-Cases befindet. Dies ist insbesondere beim Checkout-Prozess von hoher Wichtigkeit, wo über eine prominente und transparente Schritt für Schritt Navigation klar erkennbar ist, welchen Schritt der Nutzer bereits absolviert hat und welche noch ausstehen.

- **Erwartungskonformität**
  Hinter diesem Begriff verbirgt sich, dass ein Wording so präzise ist, dass der Nutzer genau weiss, wohin er klicken muss, um zu den gewünschten Elementen zu gelangen.

- **Farbe funktional einsetzen**
  Bei der visuellen Nutzerführung spielt die Farbgebung eine grosse Rolle. Dabei gilt es, aufmerksamsstarke Farben wie Rot, Orange und Gelb für Elemente mit zentraler Interaktionswichtigkeit einzusetzen, Farben wie Grün und Blau für Information oder Bestätigung

- **Intelligente Formulare**
  Hat ein Shop eine hohe Usability, konfrontiert dann jedoch den Nutzer bei Checkout mit nervigen Formularen, kann dies ein Konversionsgrund sein. Daher sind intelligente Formulare mit direktem, sprechenden Feedback ein wichtiges Detail für die User Experience

Für die konkrete Usability-Evaluierung habe ich folgende Elemente geprüft:

- klare Struktur der Landingpage
- Möglichkeit einer Navigation mittels Bildsprache auf Kategorieebene
- hohe Qualität der Suchfunktion
- intuitiv bedienbare Produktfilterfunktionen
- Möglichkeit für Cross- und Upselling
- Dialog- und Formulargestaltung auf der Höhe der Zeit

Da Usability bei hochwertiger Integration ein erheblicher Kostenfaktor ist, habe ich bei der Auswahl der E-Shop Software diejenigen bevorzugt, die bereits im Auslieferungszustand eine hohe Usability aufweisen. Die dafür notwendigen, bereits vom Hersteller durchgeführten Usability-Tests, verringern den für die Projektdurchführung notwendigen Aufwand und bieten gleichzeitig eine hohe Produktqualität.

Intelligente Suchfunktion

Die Suchfunktion ist ein Element, welches sich in der Schnittmenge von Artikelverwaltung und allgemeiner Shop-Usability befindet. Die Wichtigkeit dieses Elements kann laut Gero Lüben im E-Commerce Leitfaden gar nicht überschätzt werden: „Die Suchfunktion entscheidet über Kauf oder Nichtkauf“ [16]. Dies begründet er zunächst damit, das die Suchfunktion die am häufigste benutzte Einstiegspforte und Funktion im Shop ist. Die im Jahr 2011 von der ibi Research an der Universität Regensburg durchgeführte Befragung von Online-Händlern untermauert diese These:

E-Commerce Systeme

Evaluierungs-kriterien

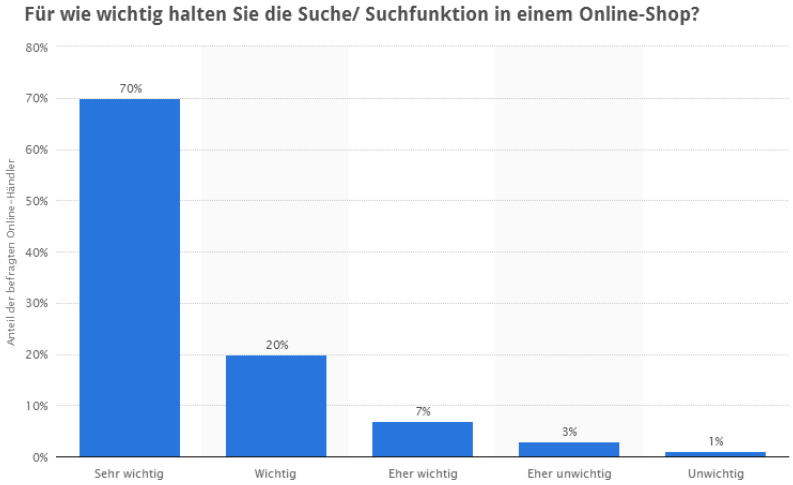


Abbildung „Wichtigkeit der Suchfunktion in einem Online-Shop“[20]

Als optimal wird eine sog. „Auto-Suggest-Suche“ bewertet, die per simpler Texteingabe mit ergänzender Dropdown / Overlaybestätigung mit nur wenig Klicks zum Ergebnis führt. Dabei „muss der User sich hier nicht sicher sein, welcher Kategorie sein gewünschtes Produkt zugeordnet ist.“, wie Gero Lüben, Geschäftsführer eines erfolgreichen Herstellers für Suchsoftware schildert [16]. Optimalerweise ist eine intelligente Suche mittels einer Autocomplete-Suche mit nachgelagerter Ergebnisseite strukturiert, die wiederum Produktfilter bietet: „Auf der Ergebnisseite muss sich dem User dann aber auch eine komfortable, der Suchanfrage angepasste Navigation eröffnen. Im Fachjargon nennt sich das Dynamische Navigation oder After-Search-Navigation. Dabei sind Filter anklickbar, mit denen die Ergebnisse verfeinert bzw. eingrenzt werden können, um so einfach das gewünschte Produkt zu finden. Das können allgemeine Filter, wie Kategorie, Hersteller oder Preis, sein, aber auch shop-individuelle Filter sollten dargestellt werden...“. Für die Evaluierung bei der Shopauswahl bedeutete dies, dass eine intelligente Suche als sehr wichtiges Kriterium bei der Entscheidungsfindung geführt wurde.

Kick-Off-Kriterien

Um die Auswahl der geeigneten E-Shop Software zu erleichtern, wurde als erstes mittels Ausschlusskriterien die Menge der zu prüfenden Shopsysteme verringert.
Nicht in Frage für das Projekt „Sunrise“ kamen Systeme, die mindestens eine der folgenden Kriterien erfüllt:

E-Commerce Systeme

Evaluierungs-kriterien

- überwiegend für den internationalen Markt konzipiert (Kerngeschäft von Sunrise: Deutschland)
- geschlossene Kaufsoftware (kein Einsicht in den Code, keine Anpassungen seitens Equinoxe möglich)
- fertige Mietsoftware (keine Anpassungen möglich)
- wenig verbreitete Nischenprodukte (nicht zukunftssicher)
- E-Shops mit mangelhafter oder unstrukturierter Dokumentation (zu hohe Produktionskosten aufgrund zu großem Aufwand für Know-How-Gewinn)
- E-Shops mit schwacher Community (Nutzen des E-Shops ohne Supportvertrag kaum möglich)

Eine Sonderolle hierbei kam Magento zu - eine E-Shop Anwendung, die vom technischen und konzeptionellen Aspekt her sehr vielversprechend ist, jedoch aus anderen Gründen als schwierig eingestuft wird:

Alexander Graf von kassenzone.de beschreibt dies in seinem Online-Artikel „Das beste Shopsystem“ so [3]:
„Bis zur Übernahme durch eBay hatte das Magento System einen extrem großen Zulauf. [...] Die riesige Community & der Bekanntheitsvorsprung waren sehr wichtige Faktoren beim Ringen um die Gunst der Shopbetreiber. Die großen Herausforderungen für die Systemanbieter besteht in der ständigen Anpassung ihrer Systeme hinsichtlich der Marktgegebenheiten und der Pflege einer schlagkräftigen Community. Beiden Herausforderungen kann Magento [...] unter der Leitung von eBay niemals gerecht werden. Die Community ist nahezu regungslos [...].“
Ich habe Magento dennoch in die Liste der engeren Wahl bei der Evaluierung, welches Shopsystem für das Projekt „Sunrise“ am besten geeignet ist, aufgenommen, da es - abgesehen von den Gefahren, die durch die Übernahme von Magento durch Ebay existieren, ein großes, leistungsstarkes und verbreitetes E-Shop System ist.
Die Ausrichtung von Magento kann zwar international bezeichnet werden, jedoch existieren Erweiterungen wie „Market Ready Germany“, die das E-Shop System um deutschlandspezifische Funktionen erweitert.

8.3. Der Idealfall - ein integriertes ERP

Bei der Beratung des Kunden legte ich ihm nahe, die Gunst der Stunde zu nutzen, um im Zuge des Relaunchs ein zugehöriges ERP einzurichten, mit dem u.a. die rein kaufmännische Seite des Shop deutlich professioneller gehandhabt werden kann als bei der Verwendung eines reinen Webshop.

E-Commerce Systeme

Kick-off-Kriterien

Im Vorfeld des Projekts habe ich eine Studie durchgeführt, bei der infrage kommende ERPs nach kundenspezifischen Kriterien tabellarisch eingestuft wurden (vgl. Abbildung „Evaluierung ERP Systeme“ im Anhang). Leider – die Hennes betreiben ihr Geschäft mit nur wenigen Mitarbeitern – hatten sie nicht ausreichend Kapazitäten frei, um diesen Teil auch noch zu bewältigen.

8.4. konkrete Evaluierung und Auswahl

Die konkrete Evaluierung wurde bis zu dem Zeitpunkt von mir durchgeführt, an dem dem Kunden eine engere Wahl zur Ansicht vorgestellt wurde. Die Bewertung der E-Shop-Software wurde wahlweise mittels Demoversionen im Netz oder lokal installierten Community-Versionen durchgeführt. Dabei wurden Testszenarien
wie z.B. Anlegen von Artikelstrukturen und Artikeln durchgespielt, sowohl im Backend wie im Frontend.
Die dabei entstandenen Bewertungsdaten waren zwar unabdingbar für die Auswahl, erheben jedoch keinen wissenschaftlichen Anspruch.
Oft musste - bedingt durch den Projektalltag in einer Agentur -
ein kurzer Eindruck für ein Teilfeature reichen.
Daher stützte ich meine Einschätzung zusätzlich auf Artikel
aus Fachforen und E-Commerce Portalen wie
Alexander Grafs Artikel „Das beste Shopsystem“
http://www.kassenzone.de/2012/03/25/das-beste-shopsystem/
Für die Evaluierung wurden folgende E-Shop-Softwares herangezogen:

1. Magento Community Edition, Version 1.8.1
   (lokale Installation)
2. Shopware Community Version 4.0.1 (lokale Installation)
3. OXID Community Version 4.5 (lokale Installation)
4. XT Commerce Version 4.0.14 (Online-Demo des Herstellers)
5. OS-Commerce Version 2.3.1 (lokale Installation)

Die Evaluierung für die Entscheidung über die verwendete E-Shop Software habe ich in folgende Bereiche unterteilt:

- Katalog/Artikelverwaltung
- Technik
- Frontend/Layout

- Marketing
- Usability / Support

wobei ich eine unterschiedliche Anzahl von Key-Features geprüft habe - so habe ich beispielsweise die Artikelverwaltung deutlich ausführlicher geprüft als das Frontend-Layout - letzteres kann leichter durch ein anderes Template ersetzt werden, Backend-Strukturen sind jedoch der strukturelle Unterbau eines Systems und daher nur mit grossem - finanziellen wie zeitlichen - Aufwand zu ändern, was bei einem kleinen Kunden wie Sunrise keinen Sinn macht.

E-Commerce Systeme

Usability und Support habe ich bei der Evaluierung aus dem Grund zusammengefasst, weil beide Bereiche eine Aussage darüber machen, wie effizient (und auch wie gerne) man mit dem jeweiligen E-Shopsystem arbeitet, sei es als Shopbetreiber oder als Kunde. Ausgangssituation beim Testing war jeweils die Community-Version der jeweiligen E-Shop Software, ohne Erweiterungen.

konkrete Evaluierung

8.5. tabellarische Evaluierung

Bei der Evaluierung habe ich Standardszenarios durchgespielt, die dem jeweiligen zu testenden Feature entsprachen, bespielsweise „ich möchte ein Produkt mit Eigenschaften aus verschiedenen Kategorien anlegen und dieses im Frontend filtern“ für die Evaluierung der Filterfähigkeit.
War dies out-of-the-Box ohne Schwierigkeiten möglich, habe ich dies mit ***, also 3 Punkten bewertet.
Eine Ausnahme hierbei war - meist bei Magento - die Situation, dass „out-of-the-box“ zwar nicht möglich war, jedoch eine Vielzahl an Community-Elementen für ebendiesen Bereich zur Verfügung standen, mit dem die jeweilige Anforderung gut umsetzbar gewesen wäre - dies habe ich ebenfalls mit 3 Punkten bewertet.

War es möglich, aber vom Handling eher hakelig oder nur durch Detailrecherche realisierbar, habe ich **, also 2 Punkte vergeben.
Bei mangelhafter Implementierung oder bei nur durch kostenpflichtige Erweiterungen umsetzbare Features habe ich mit *, also einen Punkt bewertet. Die Gesamtwertung innerhalb eines Bereichs errechnet sich durch die erreichte Punktzahl, geteilt durch die Anzahl der evaluierten Einzelemente. Die Summen der einzelnen Bereiche wurden in einer Übersicht zusammengestellt.
Dieses eher grobe Bewertungsraster lieferte Übersichtslisten, die eine schnelle Auswertung - und damit eine konkrete Entscheidungshilfe zulassen.
Beginnen wir mit der Evaluierung der Katalog / Artikelverwaltung:

8.5.1 Evaluierung - Katalog / Artikelverwaltung

konkrete Evaluierung

Evaluierung - Katalog/Artikelverwaltung

Feature	Magento	Shopware	OXID	XT - Commerce
Unterstützung von > 10000 Artikeln	***	***	***	***
Produktfilter	**	***	* +++	* +++
Produktvarianten	**	***	* +++	* +++
Intelligente Suchfunktion	**	***	**	* +++
Produktkonfigurator	**	***	**	* +++
Downloadbare Produkte	***	***	***	* +++
Integrierte Produktbewertung	***	***	***	* +++
Gesamtwertung	**2,4**	**3,0**	**2,1**	**1,3**

+++ nur per kostenpflichtigem Plugin

Fazit: Nur Shopware bietet ein vollständiges Paket an modernen Artikelverwaltungsfeatures, OXID (benötigt kostenpflichtige Plugins) und Magento (bietet sämtliche getesteten Features, ist aber von der Bedienung her zu komplex und damit für ein kleines bis mittelgrosses Projekt ungeeignet) folgen im Ranking, XT-Commerce ist in diesem Bereich von Haus aus am schlechtesten aufgestellt.

8.5.2 Evaluierung - Technik

Evaluierung - Technik

Feature	Magento	Shopware	OXID	XT - Commerce
Open Source als Plattform / Methodik	***	***	***	**
Programmiersprache PHP	***	***	***	***
Erweiterbarkeit (Plugins)	***	***	***	**
Schnittstellen	** +	***++	***+++	***++++
Zukunftssicherheit	**	***	***	*
Gesamtwertung	**2,6**	**3,0**	**3,0**	**2,2**

\+ = zahlreich verfügbar, aber überwiegend für den amerikanischen / intenationalen Markt
++ = stetig wachsende, gut dokumentierte REST-Schnittstelle
+++ = zentrales Management und Controlling aller Services per OXID efire
++++ = umfangreiches Angebot an vorinstallierten Schnittstellen

Fazit: Shopware und OXID teilen sich den ersten Platz - Magento und XT-Commerce verlieren vornehmlich deshalb, weil sie keine greifbare Zukunfts sicherheit bieten (Magento durch die Übernahme seitens Ebay,XT-Commerce durch den Status des angestaubten Systems, bei dem auch die Community stetig schwächer wird. Diese Informationen lassen sich schlecht statistisch belegen, hier spielt der persönliche Eindruck in den jeweiligen Fachforen eine grosse Rolle.

8.5.3 Evaluierung - Frontend / Layout

konkrete Evaluierung

Evaluierung - Frontend - Layout

Feature	Magento	Shopware	OXID	XT - Commerce
Qualität des Standardtemplates	**$^{++}$	***	**	**
Anpassungsfähigkeit des Templates	**$^{++}$	***	***	**
Unterstützung von Mobile Shopping	***	***	**	*
Verfügbarkeit Templates	***	*	**	**
Gesamtwertung	**2,5**	**2,5**	**2,3**	**1,8**

++ = Magento kommt hier eine Sonderrolle zu: was dem Standardtemplate fehlt, wird durch die grosse Zahl an Community-Templates wettgemacht

Shopware wäre in diesem Bereich (bedingt durch das hochwertige Standardtemplate) der klare Sieger, stolpert jedoch über die fehlende Auswahl an Templates (was sich in den vergangenen 2 Jahren stark geändert hat). Magento gleicht Schwächen im Standardtemplate durch die Community aus. OXIDs Standardtemplate ist zwar solide, aber bleibt hinter Magento und Shopware zurück - XT-Commerce wirkt von Haus aus eher bieder, kann jedoch - bedingt durch die ehemals grosse Verbreitung dieses Systems - mit einer grossen Zahl an Templates angepasst werden.

8.5.4 Evaluierung - Usability / Support

Evaluierung - Usability / Support

Feature	Magento	Shopware	OXID	XT - Commerce
Usability Backend	**	***	**	**
Usability Frontend / Bestellvorgang	***++	***	**	***
Community / Forum	**	***	**	*
Dokumentation / Wiki	***	***	**	**
Gesamtwertung	**2,5**	**3,0**	**2,0**	**2,0**

++ = hier kommt Magento erneut eine Sonderrolle zu - die hohe Usability wird durch die grosse Anzahl an qualitativ hochwertigen Templates erreicht

konkrete Evaluierung

Fazit: In diesem Bereich leistet sich Shopware keine Schwäche, sowohl das usability-getestete Frontend, die Vitalität des Forums und das gut strukturierte wie vollständige Wiki, gepaart mit dem Developer‘s und Designer‘s Guide sowie den zahlreichen Herstellertutorials lassen kaum Wünsche offen.

Magento verliert durch das zu komplexe Backend und die schwindende Community, was durch die grosse Anzahl an bereits existierenden Forenbeiträgen ausgeglichen wird. OXID gelingt es nicht, zu der Qualität von Shopware aufzuschliessen. Das Backend wirkt angestaubt und nicht immer gut strukturiert, das Frontend „funktioniert“ zwar, aber bietet keine echte User Experience. Auch die Community / das Wiki wirken z.T. unvollständig, unstrukturiert oder nicht auf dem neusten Stand.

Das Backend von XT-Commerce wirkt zwar überholt, kann jedoch gut und schnell bedient werden. Der eigentliche Bestellvorgang ist aus Usability-Perspektive gut umgesetzt - der User weiss stets, an welcher Stelle des Prozesses er sich befindet und was wo zu tun ist.
Die Herstellerdokumentation kann als „gut“ bezeichnet werden, wenn auch nicht auf dem Niveau von Shopware.

8.5.4 Evaluierung - Marketing

Magento ist das einzige System, bei dem die CMS-Funktionalität ausgereift und mächtig wirken - die übrigen Systeme bieten Basisfunktionalitäten, die kaum mehr als ein Blog sind. OXID bildet in diesem Bereich das Schlusslicht. Was die Bereiche Up- und Cross-Selling sowie SEO Optimierung angeht, so bieten dies alle der getesteten Systeme - die Unterschiede bei der Bewertung ergeben sich durch die Usability sowie

derm vom System bereitgestellten Intelligenz beim Umgang mit diesen Themen. Ein funktionierendes integriertes Newslettersystem - ohne vorherige Anpassungen - bieten nur Shopware und OXID, bei Magento ist die Einrichtung hakelig und XT-Commerce beinhaltet seit der Version 4 keinen Newsletterversand mehr.

Evaluierung - Marketing

Feature	Magento	Shopware	OXID	XT - Commerce
SEO optimiert	**	***	***	**
Integriertes Newslettersystem	**	***	***	*
Integriertes CMS	***	**	*	*
Up-Selling	***	***	**	**
Cross-Selling	***	***	**	**
Gesamtwertung	**2,6**	**2,8**	**2,2**	**1,6**

konkrete Evaluierung

8.5.5 Evaluierung - Gesamtwertung

Die Zusammenfassung - wiederum nach Gesamtsumme geteilt durch Anzahl der Bereiche gerechnet - zeigt alle Bereiche in der Übersicht.

Dabei schneidet Shopware nach Punkten am besten ab - wobei Magento dicht aufschliesst und auch OXID noch als „ordentlich“ bezeichnet werden kann, allein XT-Commerce kann in der Gesamtwertung nicht mithalten.

Evaluierung - Zusammenfassung

Feature	Magento	Shopware	OXID	XT - Commerce
Katalog/Artikelverwaltung	**2,4**	**3,0**	**2,1**	**1,3**
Technik	**2,6**	**3,0**	**3,0**	**2,2**
Frontend/Layout	**2,5**	**2,5**	**2,3**	**1,8**
Marketing	**2,6**	**2,8**	**2,2**	**1,6**
Usability / Support	**2,5**	**3,0**	**2,0**	**2,0**
Gesamtwertung	**2,52**	**2,86**	**2,32**	**1,78**

Da Magento verschiedene Kick-Off Kriterien aus Kundensicht nicht erfüllt (keine Ausrichtung auf den deutschen Markt, zu komplex in der Bedienung) und die Zukunftssicherheit nicht stabil gewährleistet ist, wurden Dr. Henne von Sunrise beim finalen Meeting zur Entschei-

dungsfindung nur OXID und Shopware bereitgestellt, jeweils in den bei der Evaluierung erwähnten Versionen.

Hierbei ergab sich, dass Dr. Henne sowohl das OXID-Standardtemplate als auch das Backend nicht zusagte.
Standardtemplate und Backend von Shopware hingegen wurden vom Kunden positiv aufgenommen, so dass Shopware als System für den Relaunch verabschiedet wurden.
Diese Entscheidung war aus Agenturperspektive willkommen, da dadurch ein Pilotprojekt mit Shopware 4 entstand, was eine detailliertere Evaluierung der E-Shop Software „Shopware 4“ ermöglichte.

konkrete Evaluierung

Was als Pressemitteilung im Juni 2012 [1] angekündigt wurde, klang ambitioniert:

- komplett refaktoriertes Backend (MVC / ExtJS 4.1)
- Datenbank-Models basierend auf Doctrine 2
- neuer Cache auf Basis von Symfony 2 / Varnish
- vollständige Event-/Hook-Unterstützung in allen Backend-Modulen
- stark optimierter Shopware Standard-Stack
- viele Verbesserungen und neue Features im Shopware Plugin-System
- hohe Code-Qualität durch testgetriebene Entwicklung mit Unit- und Selenium-Tests

Im folgenden Kapitel werde ich näher auf die einzelnen Komponenten von Shopware 4 eingehen und mich mit der Systemarchitektur auseinandersetzen. Um besser beurteilen zu können, wie ein System aufgebaut ist, habe ich mich vorweg mit einigen Grundlagen von Standardherangehensweisen bei der Erstellung eines System befasst, von denen ich einige herausgreifen möchte. Dabei möchte ich mich auf Entwurfsmuster für objektorientierte Anwendungen beschränken.

9. Entwurfsmuster - Grundlagen

Mein Ausgangspunkt bei der Betrachtung von Entwurfsmustern war das Standardwerk „Design Patterns. Elements of Reusable Object-Oriented Software“ der als „Gang-of-four“ (GoF) bekannten Autoren Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides.
Diese Autoren haben ingesamt 23 Entwurfsmuster erarbeitet, die bis heute Beachtung finden. Dabei ging es mir nicht darum, möglichst alle Entwurfsmuster zu behandeln, sondern eine Auswahl an klassischen Mustern vorzustellen.
Entwurfsmuster werden in drei Bereiche gegliedert:

Entwurfs-muster

1. **Strukturmuster**
2. **Verhaltensmuster**
3. **Erzeugungsmuster**

Dabei erfassen Strukturmuster hauptsächlich die statische Struktur von Klassen bzw. Objekten.Verhaltensmuster beschreiben, wie Objekte durch ihr Zusammenwirken ein bestimmtes Verhalten erzeugen, und Erzeugungsmuster befassen sich mit der Erzeugung von Objekten [11].

9.1. Factory Design Pattern

Das Factory Design Pattern zählt zu den „Creational Patterns“ und ist lt. GoF so definiert:
„Definiere eine Klassenschnittstelle mit Operationen zum Erzeugen eines Objekts, aber lasse Unterklassen entscheiden, von welcher Klasse das zu erzeugende Objekt ist. Fabrikmethoden ermöglichen es einer Klasse, die Erzeugung von Objekten an Unterklassen zu delegieren.“[10]
Ein Vorteil dieser Vorgehensweise ist, dass damit sowohl unspezialisierte Objekte wie auch Ausnahmefälle behandelt werden können, z.B. die Erstellung von regulären Bahn-Tickets wie auch spezielle Urlaubsangebote (die dann von den dafür spezialisierten Unterklassen erzeugt werden)

Weitere Vorteile sind:

- Die eigentliche Objekt-Erstellung ist vom Client abgetrennt, was eine Wiederverwendbarkeit des Codes ermöglicht
- Die Wartbarkeit des Codes wird vereinfacht, da die Objekt-Erstellung zentral verwaltet wird.

9.2. Das Strategy Design Pattern

Die Definition der GoF definiert das Strategy Design Pattern wie folgt: „Definiere eine Familie von Algorithmen, kapsele jeden einzelnen und mach sie austauschbar. Das Strategiemuster ermöglicht es, den Algorithmus unabhängig von ihn nutzenden Klienten zu variieren.“[10]
Eingeordnet wird das Strategy Design Pattern bei den sog. „Behavioral Patterns“. Dabei stellt das Strategy Design Pattern einen zentralen Aspekt der Softwareentwicklung in den Vordergrund: Die Veränderung der Anforderungen. Unter dem Begriff „Strategy“ versteht man in diesem Zusammenhang das Auslagern (Kapseln) eines Objekt-Algorithmus in eine sog. Strategieklasse.

Entwurfsmuster

Der sog. Context, also das logische Element, auf dass sich Eigenschaften und Methoden beziehen, kommuniziert mittels Schnittstelle mit den Strategieklassen, wodurch die Anpassungsfähigkeit des Codes erhöht wird, was der zentrale Vorteil dieses Entwurfsmusters ist: das Verhalten eines Systems kann dynamisch angepasst werden, ohne die Kernelemente jedes Mal anfassen zu müssen.

9.3. Das Command Design Pattern

Lt. GoF-Definition basiert das sog. Command Design Pattern auf diesem Prinzip: „„Kapsle einen Befehl als ein Objekt. Dies ermöglicht es, Klienten mit verschiedenen Anfragen zu parametrisieren, Operationen in eine Schlange zu stellen, ein Logbuch zu führen und Operationen rückgängig zu machen.“
„Durch das Befehlsmuster soll die Erteilung und die Auslösung eines Befehls zeitlich entkoppelt werden. Das bedeutet, dass das Erzeugen eines Befehls und seine Ausführung zu verschiedenen Zeiten stattfinden können. [...] Ein konkreter Befehl entspricht einem Methodenaufruf oder einer Folge von Methodenaufrufen auf einem Empfängerobjekt. Ein konkreter Befehl feigen wird zusammen mit der Referenz auf das Empfängerobjekt in einem auf dem Interface IBefehl basierenden Objekt gekapselt und wird einem Aufrufer übergeben.
Der Aufrufer hängt so nicht von den Details des Befehls ab, sondern nur vom Interface IBefehl.“ [11]
Anweisungsobjekte werden angesprochen wie „normale“ Objekte. „Rückgängig-“ sowie „Logging-“ Funktionen sind leicht implementierbar. Vorteile dieser Architektur liegen in der Modularität, der damit verbundenen Flexiblität sowie der Wiederverwendbarkeit.

9.4. Das Proxy Design Pattern

Das Proxy Design Pattern ist ein Vertreter der sog. „Structural Patterns“ und beinhaltet ein simples wie wirkungsvolles Prinzip:
„Kontrolliere den Zugriff auf ein Objekt mit Hilfe eines vorgelagerten Stellvertreterobjekts“. Dabei representiert der sog. Proxie das eigentliche Objekt und übernimmt Aufgaben wie Initialisierung sowie Erweiterung. Am Beispiel „Bild“ wird der Vorteil dieses Musters deutlich:
Der BildProxy übernimmt die Aufgabe, das physikalische Bild zu laden, sobald es benötigt wird. Dabei verfügt der Proxy z.B. über Informationen wie „Bildgrösse“, so dass er der Programmierung vor der eigentlichen Auslieferung des Bilds bereits Informationen über dieses liefern kann. Auch kann der Proxy Dinge wie das Zusammenführen von Bild- und eines - im Proxy festgelegten - Bildrahmens - ausführen.

Entwurfs-muster

9.5. Das Observer Design Pattern

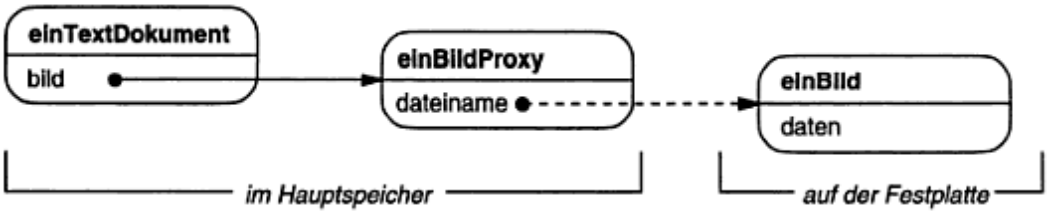


Abbildung GoF Entwurfsmuster - Proxie - Beispiel „Bild“ [40]

Das sog. „Observer Design Pattern“ lautet wie folgt: „Definiere eine 1-N Abhängigkeit zwischen Objekten, so dass die Änderung des Zustands eines Objekts dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden“ [TODO GoF].
Lt. Goll / Dausmann [TODO] enthält dieses Muster folgende Elemente:

- ein Daten haltendes Objekt (Observable / Subject), dessen Daten und damit dessen Zustand sich ändern können
- mehrere Interessenten für die gekapselten Daten (Observer), die gegebenenfalls auf Änderungen der Daten reagieren wollen.

Ein Beispiel macht dies greifbar:

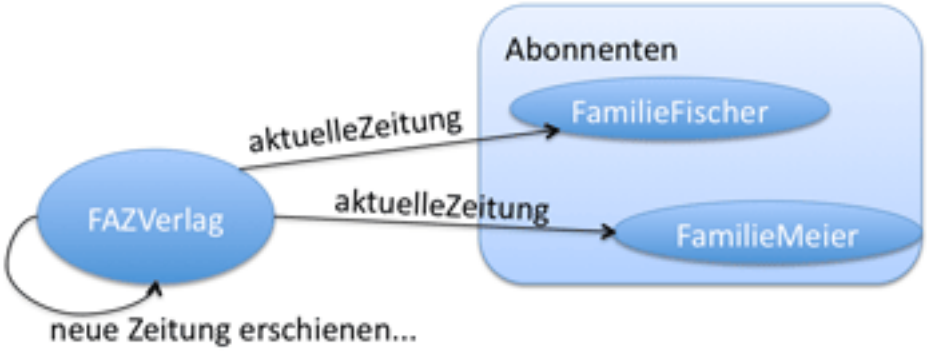


Abbildung GoF Entwurfsmuster - Observer - Beispiel „Zeitung“ [41]

Wird eine neue Zeitung (=Subject) veröffentlicht, so wird diese allen Abonnenten (=Observer) zugestellt.

Die Vorteile dieser Architektur bestehen darin, dass die Daten im Gesamtsystem konsistent bleiben, da die Observer ihren Zustand - um damit ihre Daten - bei Änderung der Daten des sog. Subjects - abgleichen. Dabei kann es mehrere Observer für ein Subject geben wie auch das Beobachten mehrerer Subjects von einem Observer. Dies dient erneut der Modularisierung und Flexibilisierung.

9.6. Aspektorientierte Programmierung

Aspektorientierte Programmierung ist ein recht junges, modernes Entwurfsmuster. Zielsetzung dieses Programmierparadigmas ist es, Komponenten von Aspekten zu trennen. Dies bedeutet, dass Code getrennt von Aspekten, die nichts mit der eigentlichen Aufgabe der Komponente zu tun hat (wie Protokollierung, Fehlerbehandlung, Persistenz, Datenvalidierung und Security, den sogenannten „Cross-Cutting Concerns“) behandelt wird. Sog. „Scattering“ (= redundanten oder zumindest ähnlichen Code-Blöcken in allen betroffenen Modulen, was zu einer schlechten Pflegbarkeit des Systems führt) sowie „Tangling“ (= Code Fragmente, die mehrere Anforderungen betreffen, jedoch in einem Modul vermischt sind) gilt es zu vermeiden.
Wichtige Begriffe der aspektorientierten Programmierung sind diese [14]:

Entwurfsmuster

- **Verantwortung.**
  Jedem Modul ist ein klarer Verantwortungsbereich zugeordnet.

- **Modularisierung.**
  Durch Vermeidung von Scattering und Tangling wird der Programmcode besser modularisiert. Dies hat zur Folge eine reduzierte Redundanz, die Erhöhung der Verständlichkeit und letztendlich eine besser zu pflegende Software.

- **Einfachheit.**
  Architektur und Entwurf können sich auf die Anforderungen der Anwendung selbst konzentrieren. Diesem Grundsatz nach Einfachheit des Entwurfs folgt auch der Ansatz des extreme Programming.

- **Wiederverwendbarkeit.**
  Aspekte sind von ihrem Ansatz weniger eng an eine Anwendung gekoppelt, als das der Fall bei einer konventionellen Programmierung ist. Daher ist aspektorientierter Code im Allgemeinen gut wiederverwendbar.

- **Stabilität.**
  Dadurch, dass Crosscutting Concerns bei der Implementierung weiterer anwendungsspezifischer Funktionalität weniger schnell übersehen werden, wird der Code insgesamt stabiler.

- **Kosten.**
  Einfachheit, Wiederverwendbarkeit und Stabilität wirken sich nachhaltig positiv auf die Kosten aus. Zudem kann das System früher genutzt werden, und damit amortisieren sich auch die Kosten der Entwicklung früher.

Diesen Vorteilen stehen aber auch durchaus Praxis-bezogene Nachteile gegenüber:

- **Debugging.**
  Aufgrund der Trennung von Code- und Prozessstrukur kann dem Kontrollfluss in einem Programm nur mit Schwierigkeiten gefolgt werden. Dies ist vergleichbar mit der dynamischen Bindung in der objektorientierten Programmierung.

- **Performance.**
  Beim Weben zur Laufzeit (dynamisches Weaving) sind Reduzierungen in der Performance nachzuvollziehen.

- **Kapselung.**
  Durch die AOP wird das objektorientierte Prinzip der Kapselung gebrochen, da eine Klasse nicht mehr ihr gesamtes Verhalten unter eigener Kontrolle hat.

- **Kopplung.**
  Sog. Pointcuts (= Auswahl einer Menge von Elementen) werden überwiegend über Namen von Klassen, Methoden, Variablen oder auch Paketen definiert. Das hat eine Kopplung von Komponenten und Aspekten zur Folge und führt u.a. zur Forderung nach einer strikten Einhaltung von Namenskonventionen.

- **Disziplin.**
  Die Anwendung der AOP setzt die Einhaltung von Software-Standards nicht nur im Hinblick auf Namenskonventionen voraus, was in größeren Entwicklungsteams u. U. mit Schwierigkeiten verbunden ist.

10. Shopware - eine Analyse

Im folgenden möchte ich die zugrundeliegenden Architekturen von Shopware untersuchen. Die Untersuchung der nachfolgenden Elemente geschieht sowohl aus Entwickler- wie aus Agenturperspektive und dient der näheren Einschätzung wie auch des Wissenszuwachses bei den beteiligten Technologien.

10.1. Systemarchitektur

10.1.1 Möglichkeiten und Grenzen der Systemarchitektur-Analyse

Shopware System-architektur

Da Shopware mit mächtigen, komplexen Frameworks realisiert ist, die - jedes für sich 1-2 Jahre Erfahrung benötigen, um diese professionell einsetzen zu können, habe ich die Analyse auf das jeweilige Kernkonzept eines Frameworks beschränkt. Weder eine Beurteilung der Qualität des Codes bei der eigentlichen Implementierung noch eine Einschätzung, „aus welcher Welt“ Elemente innerhalb des Shopware-Codes stammen ist mit meinem Vorwissen möglich (Spätestens bei der Implementierung des Backend-Plugins musste ich feststellen, dass Shopware Elemente der beteiligten Frameworks nach Bedarf modifiziert und erweitert, was ein Verständnis des Codes erschwert).
Von grossen Interesse war jedoch Einschätzung, ob die verwendeten Komponenten zukunftsgerichtet und mächtig sind, also über die Anwendung innerhalb Shopware für die Agentur Equinoxe von Interesse sind.

Für jedes beteiligte Framework habe ich eine kurze Zusammenfassung des Basiskonzepts sowie eine Auswahl der jeweiligen Kernkomponenten erarbeitet.

10.1.2 Shopware 4 - eine Übersicht

Shopware 4 basiert grundsätzlich auf objektorientiertem PHP, also 5.3.x oder höher. Dabei wurden beim Redesign des Major Releases, Version 4, ein Grossteil der Funktionalitäten auf Frameworks verteilt. Diese bieten zum einen durch ihre jeweilige Spezialisierung leistungsstarke Features, wie auch eine Vereinheitlichung bei der Vorgehensweise. Letzteres erlaubt eine besseres Update-Handling und dient als Basis für Unit-Tests. Die beteiligten Frameworks sind diese:

Shopware System-architektur

- **Zend Framework 2** (bekanntest PHP Framework)
- **Enlight** (E-Commerce-Erweiterung von Zend)
- **Symfony 2** (PHP Framework für Controlling und Caching)
- **Doctrine 2** (PHP Framework für Models)
- **Smarty** (PHP Framework, vorwiegend für Frontend-Views)
- **extJS** (PHP Framework, vorwiegend für Backend-Views)

Bevor ich auf die Spezialisierung und Wechselwirkung der Komponenten untereinander eingehe, so möchte ich die einzelnen Frameworks zunächst „vorstellen“

10.1.3 Zend Framework

Das Zend Framework ist eines der „grossen“ PHP Frameworks für die Erstellung von Web-Apps und ist seit 2007 auf dem Markt. In der vorliegenden Version 2 basiert dieses strikt auf dem MVC Prinzip (Model, View, Controller) und profitiert von den Features von PHP 5.3.x:

- Namespaces
- neues Plugin-System
- bessere Performance

Im folgenden verwende ich die Bezeichnung „Zend Framework“ für das Zend Framework in Version 2.
Es ist - bedingt durch seine rein objektorientierte Architektur - nicht abwärtskompatibel. Die Funktionsweise des Zend Frameworks innerhalb dieser Bachelorthesis eingehend zu analysieren, würde den Rahmen sicherlich sprengen. Wichtig ist jedoch ein Grundverständnis des Zend-Frameworks.
Dieses beinhaltet eine objektorientierte Komponentensammlung für wiederkehrende Aufgaben wie Datenbankabfragen, Verarbeitung von Templates, Formularerstellung, Authentifizierung und Sicherheitsaspekt und vieles mehr. Es zeichnet sich dadurch aus, dass es als MVC-Framework angelegt ist, Komponenten jedoch auch einzeln verwendet werden können [7].

Sprich: Entwickler können die Teile des Frameworks einsetzen, die benötigt werden und das Framework mit anderen Komponenten kombinieren. Dies ist ein wichtiger Aspekt, gerade wenn es darum geht, eine Systemarchitektur zu schaffen, bei denen hochspezialisierte Frameworks zu einem grossen Ganzen verbunden werden sollen.

Es gehört zu den „grossen“ PHP Frameworks, welches zwar unter der Schirmherrschaft des Unternehmens Zend steht, jedoch mittels Mitwirkung von Hunderten von Freiwilligen entstanden ist. Es ist - mittels des Referenzhandbuchs - gut dokumentiert und ist weltweit stark verbreitet, was zahlreiche Forenbeiträge zum Umgang mit dem Framework mit sich bringt. Die Codequalität ist - lt. Zend - durch die Verwendung von automatisierten Test mittels PHPUnit auf einem messbar hohen Niveau angesiedelt. Der grosse Funktionsumfang lässt eine Vielzahl von Anwendungen zu, was jedoch die Einarbeitungszeit erhöht.

Struktur

Zend organsiert seine Klassen mittels folgender Namenskonvention: „ein Klassenname beginnt mit ‚Zend‘ und trägt den Namen des Verzeichnisbaums, in dem diese liegt“. Dies bedeutet, dass die Klasse *Zend_Controller_Front* im Verzeichnis *Zend/Controller/Front* zu finden ist. Komponenten, deren Name mit „ZendX“ oder „ZendL“ beginnt, werden nicht offiziell von Zend unterstützt - hierbei handelt es sich um Elemente, die von der Zend-Community erstellt wurden und - sofern sich diese bewährt haben - in den Zend-Core aufgenommen werden [7].

Module

Dem Begriff „Modul“ kommt im Zend Framework eine besondere Rolle zu - abweichend von der Version 1 ist das gesamte Framework in Version 2 modularisiert. Module werden nicht automatisch initialisiert, sondern müssen aktiv „angemeldet“ sein - dies schafft eine hohe Flexibilität und bietet Performancevorteile. Ein Modul kann sowohl PHP Code (unter Verwendung des MVC Prinzips) enthalten wie auch Javascript oder Grafiken. Dies wurde bewusst so konzipiert, um möglichst grosse Freiheit bei der Verwendung von Modulen zu haben. Module dienen demnach der Eigen-Organisation des Frameworks sowie dessen Erweiterung [23].
Evan Coury, Autor des Modulsystem, formuliert es so: „A re-usable piece of functionality that can be used to construct a more complex application.“ [17]
Aus Entwicklersicht stellt im Zend Framework 2 ein Modul einen Ordner eines Namespaces dar, mit der zugehörigen Modul-Klasse.

```php
namespace Album;

use Zend\ModuleManager\Feature\AutoloaderProviderInterface;
use Zend\ModuleManager\Feature\ConfigProviderInterface;

class Module implements AutoloaderProviderInterface, ConfigProviderInterface
{
    public function getAutoloaderConfig()
    {
        return array(
            'Zend\Loader\ClassMapAutoloader' => array(
                __DIR__ . '/autoload_classmap.php',
            ),
            'Zend\Loader\StandardAutoloader' => array(
                'namespaces' => array(
                    __NAMESPACE__ => __DIR__ . '/src/' . __NAMESPACE__,
                ),
            ),
        );
    }

    public function getConfig()
    {
        return include __DIR__ . '/config/module.config.php';
    }
}
```

Abbildung „Zend Modul Code Beispiel“ [39]

Für das Arbeiten mit Modulen stellt das Zend Framework folgende Elemente bereit:

Module Autoloader

(verantwortlich für das Auffinden und Laden von Modulen)

Module Manager

(führt modulspezifische Events aus, wobei das Verhalten eines Moduls stark von den jeweils registrierten Listenern abhängt - diese können für jedes beliebige Event des Moduls registriert werden)

Als hervorhebenswert ist zu erwähnen, dass der Modulmanager nicht die Aufgabe des Modul-Ladens übernimmt, sondern nur einen Event auslöst, der von Listenern verschiedener Klassen gehört werden kann, was Flexibilität beim Ladevorgang sowie bei der Systemarchitektur schafft. Unter http://modules.zendframework.com/ kann man eine Liste der Modulautoren sowie deren Module finden.

Events

Das Eventsystem basiert auf der Kernidee, dass Hooks nicht automatisch bereitgestellt und ausgeführt werden, sondern - je nach Bedarf - mittels sog. Listener an Ablaufpunkte des Frameworks angehängt werden können. Dies ermöglicht eine hohe Flexibilität beim Verwalten und Ausführen von Events.
Intern basiert das Eventsystem auf dem sog. „Observer Pattern“.
Der praktische Nutzen dieser Architektur liegt darin, dass Ereignisse auszulöst werden können, die von anderen Klassen ausgewertet werden können - dies bietet ein Instrument für aspektorientierte Programmierung.

Dependency Injection

Dependency injection ist ein Begriff, der im Kontext „Objektorientierung“ von Bedeutung ist. Dabei bietet Dependency Injection eine Lösung für die Situation, dass beim Instantiieren eines Objekts einer Klasse (A) ein Objekt einer anderen Klasse (B) erzeugt und A zugewiesen werden soll. Fabien Potencier, CEO bei Sensio Labs und führender Kopf des PHP Frameworks Symfony, beschreibt dies in seinem Essay „What is Dependency Injection ?“ [9] an einem Beispiel:
Dabei geht er von zwei Klassen, „User“ und „SessionStorage“ aus.

```php
class SessionStorage
{
  function __construct($cookieName = 'PHP_SESS_ID')
  {
    session_name($cookieName);
    session_start();
  }

  function set($key, $value)
  {
    $_SESSION[$key] = $value;
  }

  function get($key)
  {
    return $_SESSION[$key];
  }

  // ...
}
```

Abbildung „Dependency Injection - Klasse SessionStorage“

```php
class User
{
  protected $storage;

  function __construct()
  {
    $this->storage = new SessionStorage();
  }

  function setLanguage($language)
  {
    $this->storage->set('language', $language);
  }

  function getLanguage()
  {
    return $this->storage->get('language');
  }

  // ...
}
```

Abbildung „Dependency Injection - Klasse User“

Sobald erforderlich ist, dass innerhalb der Klasse „User“ eine Instanz von „SessionStorage“ benötigt wird, kann dies zwar auf dem Weg der hardgecodeten Instantiierung von „SessionStorage“ geschehen:

```php
class User
{
  function __construct($sessionName)
  {
    $this->storage = new SessionStorage($sessionName);
  }

  // ...
}

$user = new User('SESSION_ID');
```

Abbildung „SessionStorage innerhalb der Klasse User“

Dies führt jedoch spätestens zu Problemen, wenn anstelle von „SessionStorage“ eine andere, z.B. von dieser Klasse abgeleitete und erweiterte Klasse als Sessionklasse verwendet werden soll - die Klasse „User“ müsste in diesem Fall angepasst werden.

Da dies mindestens bei grossen Projekten zu schwer wartbaren Code führt, sieht eine „Best practise“ in diesem Fall so aus:

```php
class User
{
  function __construct($storage)
  {
    $this->storage = $storage;
  }

  // ...
}
```

```php
$storage = new SessionStorage('SESSION_ID');
$user = new User($storage);
```

Abbildung „SessionStorage mit Dependency Injection“

Mittels der Variable „$storage“, die eine Instanz von SessionStorage enthält, wird der Klasse „User“ als Konstruktorargument mitgegeben.

Auf der nächsten Seite sieht man eine Übersicht möglicher Varianten von Dependency Injection - sämtliche Varianten haben eines gemeinsam: sie vermeiden ein Instantieren von Objekten in einer Klasse eines anderen Typs - diese wird ausserhalb der Klasse erzeugt und dann zugewiesen.
Damit ist das Prinzip klar und der Vorteil evident.
Das Zend Framework bietet mittels der Klasse „Zend_Di“ eine Reihe von Funktionalitäten, die Dependency Injection ermöglichen und standardisieren. Dies erhöht die Wartbarkeit und Skalierbarkeit eines Projekts.

Service Manager

Eine weiterer wichtiger Begriff ist der Service Manager.
Dieser übernimmt die Rolle des Baumeisters und steuert bei der Instantierung von Elementen neben der eigentlichen Objekterstellung auch das automatisierte Handling von Konfigurationen und Abhängigkeiten. Dabei muss er nicht jedes Mal durch die jeweiligen Klassen parsen, sondern verwendet - via ServiceLocatorInterface - eine Art Klassenkatalog, was einen Performancezuwachs gegenüber des klassischen Dependency Injection Modells darstellt. Auch stellt dieser sicher, dass ein Element nur einmal initialisiert wird [18].
Am Beispiel „Translator“ wird dies deutlich:

$e->getApplication()->getServiceManager()->get(‚translator‘);

Jeder weitere Aufruf des Translators würde die beim ersten Aufruf erzeugte Instanz zurückliefern.

- Constructor Injection:

```
class User
{
  function __construct($storage)
  {
    $this->storage = $storage;
  }

  // ...
}
```

- Setter Injection:

```
class User
{
  function setSessionStorage($storage)
  {
    $this->storage = $storage;
  }

  // ...
}
```

- Property Injection:

```
class User
{
  public $sessionStorage;
}

$user->sessionStorage = $storage;
```

Abbildung Dependency Injection - Übersicht

Implementierung von MVC

Die Implementierung von MVC in Zend möchte nur kurz ansprechen, da Shopware 4 diese Architektur mittels der Frameworks Symfony (Routing, Controlling), Doctrine (Model) sowie der Template-Frameworks Smarty und extJs (View) umsetzt.
Bereits im Kernsytem des Zend Frameworks werden zwar Klassen für Routing und Controlling bereitgestellt, der Bereich Model wurde hingegen offen gehalten, um eine möglichst grosse Flexibilität bei der Implementierung zu gewährleisten [19].

Zum besseren Verständnis der Shopware-Systemarchitektur möchte ich jedoch einen Begriff herausstellen, den Shopware von Zend übernommen hat: der Dispatcher. Die Zend Dokumentation definiert dies wie folgt: „Dispatching ist der Prozess, den Controller und die Aktion aus dem Request Objekt abzufragen und auf eine Controller Datei (oder Klasse) und eine Aktionsmethode in dieser Controller Klasse abzubilden. Wenn der Controller oder die Aktion nicht existieren, ermittelt es den zu verarbeitenden Standard Controller und Aktion.“ [47]

10.1.4 Die Rolle von Enlight

Enlight ist ein von der Shopware AG entwickeltes, dann unter einer BSD-Lizenz als Open Source Produkt freigegebenes Framework speziell für den E-Commerce Bereich. Es basiert auf dem Zend Framework 2 und wurde laut Aussage des Herstellers auf Performance und Erweiterbarkeit hin optimiert.
Dabei wurden vorwiegend diese Bereiche mit Funktionen erweitert:

- Dispatcher
- Eventsystem
- Hooks
- Proxie-Generierung

Shopware System-architektur

Auf diese Begriffe werde ich später im Kapitel „Systemarchitektur“ näher eingehen. Stefan Hamann von der Shopware AG hierzu: „Der große Unterschied liegt im Plugin-System. Hier gibt es zwei zentrale Möglichkeiten, Enlight-Applikationen bzw. Shopware anzupassen. Neben einem Event-System, welches die Ausführung eigener Codes beifest definierten Ereignissen (zum Beispiel Post-Dispatch) ermöglicht, gibt es ein Hook-System, mit dem man jedes Objekt und jede Methode innerhalb des Programms modifizieren kann, ohne die Objekte manuell erweitern zu müssen.“ [37]

Die eingangs ambitionierten Pläne hinsichtlich Enlight („Etablierung als eigenständiges E-Commerce Framework am Markt, spannende Unterprojekte wie „Enlight Scrum“, „Enlight Sales“ und das übergeordnete „Enlight Corporate Information System“) konnten jedoch nicht umgesetzt werden - es blieb bei der alleinigen Verwendung innerhalb Shopware 4. Dies wurde u.a. durch das Erscheinen von Symfony 2 (und dessen Verwendung in Shopware) bedingt, so dass Teile von Enlight überflüssig wurden und die Weiterentwicklung als eigenständiges Framework nach außen hin 2013 eingestellt wurde.
Für die Verwendung in Shopware wird Enlight jedoch nach wie vor weiterentwickelt.

10.1.5 Symfony 2

Symfony gilt - spätestens seit der Veröffentlichung der Version 2.1 im September 2012 als das zweite „grosse PHP Framework neben Zend“, vor dem Aspekt Leistungsfähigkeit und Flexibilität betrachtet. Es mit PHP 5 umgesetzt, ist MVC-basierend (wobei es Symfony offen lässt, ob man mit diesem Konzept arbeiten möchte oder nicht) und gilt als nahezu beliebig konfigurierbar.

Fabien Potencier, der führende Kopf des Projekts Symfony, beschreibt es in seinem Essay „What is Symfony2 ?“ so: „Symfony2 is a reusable set of standalone, decoupled, and cohesive PHP components that solve common web development problems. Then, based on these components, Symfony2 is also a full-stack web framework. “[8]
Es versteht sich also als Sammlung unabhängiger Einzelkomponenten wie auch als Web-Framework.

Fabien Potencier betont eine weitere grundlegende Eigenschaft von Symfony: „Symfony2 is an HTTP framework, it is a Request/Response framework. That‘s the big deal. The fundamental principles of Symfony2 are centered around the HTTP specification.“

Shopware Systemarchitektur

Was den modularen Gebrauch von Symfony-Komponenten und deren Einsatz in Wechselwirkung mit anderen Frameworks betrifft, geht Fabien Potentier sogar so weit, die Netzgemeinde aktiv dazu einzuladen, ihr eigenes Framework auf der Basis von Symfony Komponenten aufzubauen [7]

Im folgenden werde ich den Begriff „Symfony“ für das Framework in Version 2 verwenden. Dieses ist von vorne herein für die Integration mit anderen Frameworks ausgelegt, so wird z.B. auf Datenhaltungsebene nativ das ORM von Doctrine unterstützt. Weitere Komponenten, deren Integration seitens Symfony unterstützt werden, sind Monolog, Composer, Doctrine, Propel, Assetic, Twig und Swiftmailer. Im Umkehrschluss lässt sich Symfony ebenfalls in andere Frameworks integrieren - so verwendet der Major Release von Drupal in Version 8 Funktionalitäten von Symfony.

Symfony unterstützt - ähnlich wie Ruby on Rails - Scaffolding, d.h. automatisiertes Anlegen von Code oder Strukturen auf der Grundlage von Konfigurationsdateien. In Symfony spielen Konfigurationsdateien generell eine grosse Rolle.

Vom Niveau her ist Symfony als eher schwergewichtig zu bezeichnen, analog zum Zend Framework. Es beinhaltet umfangreiche Funktionen sowie ein mächtiges, flexibles Konzept - benötigt jedoch eine längere Einarbeitungszeit. Hierfür wird herstellerseitig das „Symfony Book“ [46] (Grundlagenwissen über die einzelnen Komponenten) sowie das „Symfony Cookbook“ [45] (Tutorials zur Implementierung mit konkreten Anwendungsbeispielen) bereitgestellt. Beides steht online zur Verfügung und wird regelmässig aktualisiert - man kann sich beide Bücher tagesaktuell als PDF herunterladen.

Symfony deckt folgende Bereiche ab:

- Frontcontroller
- Bootstrap
- Routing
- MVC
- Configuration
- Cache
- Forms
- Autoloading (von Klassen)
- Service Container (zum Aufbau von Web-Services)
- Security
- Persistence
- Dependency Injection
- Templates

Symfony

Ich möchte einige Elemente herausgreifen, die wahlweise intelligent und zukunftsweisen gelöst sind oder in direktem Bezug zu der Integration in Shopware stehen. Symfony setzt bei Grundlagen an, die im Umgang mit PHP gerne als „gegeben“ akzeptiert werden und bietet Alternativen bei der Implementierung.

Ich möchte mit dem Konzept von Request/Response beginnen: PHP 5 „verpackt“ Requests (also Anfragen an den Webserver) in globale Variablen ($_SERVER / $_GET) und bietet mittels header() die Möglichkeit, Antworten des Servers (im folgenden als Response bezeichnet) an den Client ausliefern zu lassen. Symfony bietet für Requests eine die HTTP Foundation angelehnte Klasse, die alle relevanten Informationen in einer Instanz dieser Klasse bündelt.

```php
use Symfony\Component\HttpFoundation\Request;

$request = Request::createFromGlobals();

// the URI being requested (e.g. /about) minus any query parameters
$request->getPathInfo();

// retrieve GET and POST variables respectively
$request->query->get('foo');
$request->request->get('bar', 'default value if bar does not exist');

// retrieve SERVER variables
$request->server->get('HTTP_HOST');

// retrieves an instance of UploadedFile identified by foo
$request->files->get('foo');

// retrieve a COOKIE value
$request->cookies->get('PHPSESSID');

// retrieve an HTTP request header, with normalized, lowercase keys
$request->headers->get('host');
$request->headers->get('content_type');

$request->getMethod();          // GET, POST, PUT, DELETE, HEAD
$request->getLanguages();       // an array of languages the client accepts
```

Abbildung „Symfony - Grundlagen - Request“ [42]

Dabei sind viele nützliche Dinge wie die Überprüfung, ob der Client den Request von einer verschlüsselten Seite ausgeführt hat (Methode „isSecure()“) möglich. Analog hierzu exisiert eine Response-Klasse, die eine PHP-Instanz des HTTP Response darstellt und (seit Symfony 2.4) sogar zulässt, dass HTTP Status Codes gesetzt werden können.

```php
use Symfony\Component\HttpFoundation\Response;
$response = new Response();

$response->setContent('<html><body><h1>Hello world!</h1></body></html>');
$response->setStatusCode(Response::HTTP_OK);
$response->headers->set('Content-Type', 'text/html');

// prints the HTTP headers followed by the content
$response->send();
```

Abbildung „Symfony - Grundlagen - Response [43]

Diese grundlegenden Elemente können losgelöst von restlichen Framework verwendet werden und bieten zudem Funktionalitäten wie „File Upload“ und „Session Handling“. Die Liste der definierten Objekte der HTTP Foundation sowie deren Methoden las sich sehr anspre-chend (*http://symfony.com/doc/current/components/http_foundati-on/introduction.html)*.

Der wohl grösste Vorteil der HTTP Foundation liegt m.A. nach darin, dass das Reponse-Objekt so viele Formate unterstützt, so z.B. dass - möchte man selbst einen Webservice aufbauen - wahlweise JSON oder SOAP in einem Response zurückgeliefert werden kann.
Auch wirkt der Ansatz, „holprige“ Kernelemente von PHP mit einem neuen, durchdacht wirkenden Ansatz zu ersetzen, gelungen.
Die grundlegende Architektur von Symfony beruht auf einem Front Controller, also einer einer Datei, die alle Requests entgegen nimmt. Dieser gibt die Anfrage an den Symfony Kernel weiter, der mittels Routing den passenden Controller ansteuert und damit den gewünsch-ten Content generiert / ausliefert. In der Übersicht kann man dies gut erkennen:

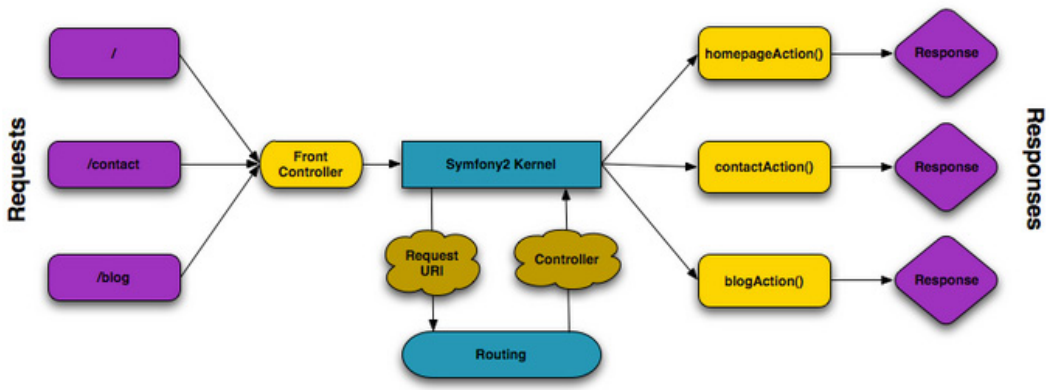


Abbildung „The Symfony Application Flow“ [44]

Bevor ich weiter auf das Routing-Konzept eingehe, möchte ich mich kurz den eingangs erwähnten Konfigurationsdateien zuwenden.

Symfony 2 unterstützt das Format YAML, ein leichtgewichtiger XML-“Verwandter“. Mittels dieser Dateien können beliebige Funktionen in Symfony parst diese Dateien und generiert Funktionscode daraus - mit allen Elementen, die dazu benötigt werden, um etwas auszuführen.

Routing

Ein Beispiel hierfür ist das Registrieren eines Routings, also der Intelligenz beim Empfangen und Auswerten von Requests - Symfony bindet das benötigte Package „Routing“ ein, initialisiert eine Instanz der Klasse „RouteCollection“ und fügt dem Routing-Handling die erzeugte Instanz hinzu. Sprich: Symfony hat automatisiert Code geschrieben, der dies ausführt und als Datei hinterlegt.
An einem Codebeispiel wird dies deutlich - der Pfad „/contact“ wird in der Routing-Konfigurationsdatei eingetragen:

Symfony

YAML XML PHP

```yaml
# app/config/routing.yml
contact:
    path:     /contact
    defaults: { _controller: AcmeDemoBundle:Main:contact }
```

Abbildung „Symfony - Routing - YAML“ [45]

Zum besseren Verständnis hier die XML-Entsprechung dieser Datei:

YAML XML PHP

```xml
<!-- app/config/config.xml -->
<?xml version="1.0" encoding="UTF-8" ?>
<routes xmlns="http://symfony.com/schema/routing"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://symfony.com/schema/routing
        http://symfony.com/schema/routing/routing-1.0.xsd">

    <route id="contact" path="/contact">
        <default key="_controller">AcmeDemoBundle:Main:contact</default>
    </route>
</routes>
```

Abbildung „Symfony - Routing - XML“ [46]

Der Eintrag in der Routing-Konfigurationsdatei erzeugt auf PHP Ebene diesen Code:

YAML XML PHP

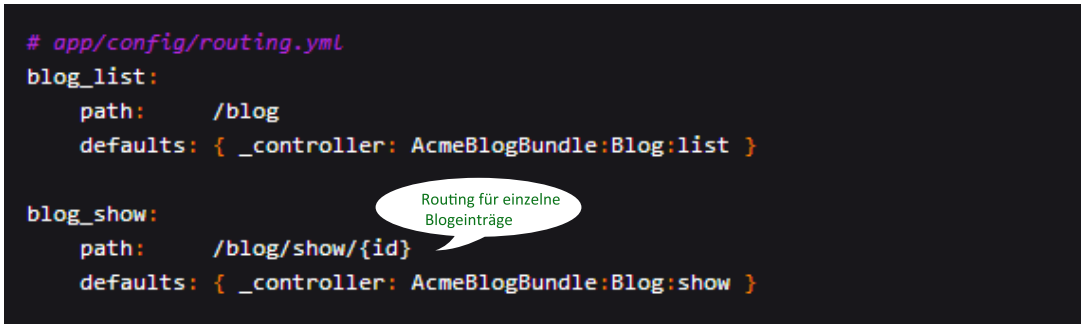
```php
// app/config/routing.php
use Symfony\Component\Routing\Route;
use Symfony\Component\Routing\RouteCollection;

$collection = new RouteCollection();
$collection->add('contact', new Route('/contact', array(
    '_controller' => 'AcmeDemoBundle:Main:contact',
)));

return $collection;
```

Abbildung „Symfony - Routing - PHP“ [47]

Die Abbildung „Symfony - Routing - Varianten“ zeigt die Einrichtung eines Routings sowohl für eine Liste wie auch für einen einzelnen Eintrag, der mittels Übergabe der jeweiligen ID angesteuert wird.




```
# app/config/routing.yml
blog_list:
    path:     /blog
    defaults: { _controller: AcmeBlogBundle:Blog:list }

blog_show:
    path:     /blog/show/{id}
    defaults: { _controller: AcmeBlogBundle:Blog:show }
```

Abbildung „Symfony - Routing - Varianten“ [48]

Eine besondere Stärke dieser Architektur liegt darin, dass mittels Konfigurationsdateien unkompliziert Anweisungen zwischen verschiedenen Anwendungen ausgetauscht werden können, was in einem verteilten System mit Anbindung an z.B. mobile Geräte von erheblichen Vorteil ist. Auch werden Code-Fehler minimiert, da dieser - eine korrekte Konfigurationsdatei vorausgesetzt - vom System erzeugt wird. Deutlich wird auch, dass XML deutlich mehr Code benötigt, um die gleiche Information wie mit YAML zu transportieren, was einen Performance-Vorteil bei YAML darstellt, welches - ähnlich wie JSON - für die leichtgewichtige Übermittlung von strukturierten Datenpaketen im Webbereich eingesetzt wird.
Die Routing-Komponente kann standalone, also ohne Einbindung des gesamten Frameworks, verwendet werden.
Sämtliche Routingdefinitionen sind in einer einzigen Datei erfasst, typischerweise in der „app/config/routing.yml“.

Controller

Controller sind in Symfony - nicht zu verwechseln mit dem Front-Controller, den es nur einmal gibt - dafür zuständig, dass vom Request angeforderte Elemente gefunden (gegebenenfalls vorher generiert) und zurückgegeben werden.
Der Ablauf beim Controlling [38] ist dieser : “

1. *Each request is handled by a single front controller file [...] that bootstraps the application*
2. *The Router reads information from the request (e.g. the URI), finds a route that matches that information, and reads the _controller parameter from the route*
3. *The controller from the matched route is executed and the code inside the controller creates and returns a Response object*
4. *The HTTP headers and content of the Response object are sent back to the client. “*

Das Mapping, mit dem festgelegt wird, welcher Controller für welche URL zuständig ist, wird mittels YAML-Konfigurationsdateien (wie im vorherigen Abschnitt gezeigt) festgelegt.

10.1.6 Datenbankstruktur

Shopware ist von Haus aus für die Verwendung von mySQL ausgelegt. Die Datenbankstruktur ist extrem aufgeräumt und intutitiv.
Durch konsequentes Naming gliedert sich die Datenbank
in folgende Bereiche (unter Verwendung des jew. Tabellenpräfixes)

Datenbank-
struktur

s_core (Kernsystem)
s_articles (alles mit Artikelbezug)
s_blog (Blog)
s_campaigns (Marketing, Einkaufswelten)
s_cms (CMS)
s_emotion (mit Shopware 4 eingeführte „Emotion“-Elemente)
s_export (Export)
s_filter (Filterfunktionen)
s_media (Medienverwaltung)
s_order (Bestellungen)
s_plugins (Plugins)
s_premium (Premium, Vorzugsbehandlung)
s_search (intelligente Suche)
s_statistics (Statistik)
s_user (User)

Dabei werden ein grosser Teil der Tabellen automatisiert von Doctrine erzeugt, das Framework, welches bei Shopware für die Datenhaltung, also das „Model“ zuständig ist - darauf werde ich im nächsten Kapitel gesondert eingehen.

Ich habe mich im Datenmodell von Shopware nach kurzer Zeit gut zurechtgefunden.

10.1.7 Das Model - Doctrine ORM

Ich möchte nun den Bereich „Symfony“ verlassen, auch wenn dieser keinesfalls umfassend behandelt wurde - Symfony kann vieles mehr - das Konzept von Routing und Controlling sollte klar geworden sein.
Wo Symfony ohnehin nicht „das Rad neu erfinden wollte“, sondern lieber mit einem anderem spezialisierten Frameworks zusammenarbeiten wollte, ist der Bereich Model. Dieses ist - gemäss MVC-Paradigma - für die Datenhaltung und Geschäfslogik zuständig.

Doctrine‘s ORM, im folgenden der Einfachkeit halber als Doctrine bezeichnet, verfolgt einen Ansatz, der objektorientierten PHP Code mit einer Datenbank zu einem grossen Ganzen verschmilzt.

Doctrine

Ein Model ist gemäss Doctrine‘s Definition ein Gebilde aus einer Datenbanktabelle, deren Attribute sowie zugehörige Methoden.
Diese klassische objekt-relationale Architektur, ein sogenannter „Object-Relational Mapper (ORM)“, bietet Vorteile, wenn man sich beim Entwickeln durchgängig auf PHP Ebene bewegen möchte, ohne echtes SQL zu schreiben oder sich um die Verwaltung der beteiligten Datenbanktabellen zu kümmern.
Doctrine übernimmt sowohl dies wie auch die Verwaltung der Beziehungen untereinander - letzteres wird jedoch nur durch einen Mehraufwand beim Anlegen der eigentlichen Models erreicht.
Die Abbildung „Doctrine - Definition Model - Auszug“ (S. 56) macht dies deutlich (auch wenn es sich hierbei um eine Integration von Doctrine innerhalb Shopware handelt).
Den Kommentare bei der Definition eines Models kommt eine Sonderrolle zu - diese dienen der Definition der „Metadata“ eines Attributs, also Datentyp sowie weitere Attribute wie Defaultwert, Erlauben von Null etc. Auch können JOINS zu anderen Models mit diesen Metadaten eingerichtet werden.
Der sog. „EntityManager“ von Doctrine steuert die Persistenz der Daten, legt also fest, bis zu welchem Punkt Daten im Arbeitsspeicher der PHP Anwendung gehalten werden, wann diese fest in die Datenbank geschrieben werden und ab wann diese Daten im Gesamtsystem verfügbar sind. Einmal eingerichtet, bietet es nützliche Zusatzfunktionen beim Liefern von Datensätzen wie Paging, Sicherheitsfeatures wie „injection protection“ sowie vorgefertigte Funktionen für viele Standardsituationen.
Die Dokumentation von Doctrine wirkt - im Vergleich zu Zend und Symfony zwar vollständig, aber wenig ansprechend - es macht sich doch bemerkbar, dass es sich nicht um ein „View“-Framework handelt ;-)
Konzeptionell versäumt die offizielle Doctrine-Website, darauf einzugehen, was die eigentlichen Stärken von Doctrine‘s ORM sind - es existiert ein „Getting Started“, der Rest ist technische Dokumentation und hat API-Charakter.

Doctrine

```php
namespace Shopware\CustomModels\Product;

use Shopware\Components\Model\ModelEntity;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\ORM\Mapping as ORM;
use Symfony\Component\Validator\Constraints as Assert;

/**
 * @ORM\Entity
 * @ORM\Table(name="s_articles_ebook_KNV")
 */
class Product extends ModelEntity
{
    /**
     * @var integer $id
     *
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="IDENTITY")
     */
    private $id;

    /**
     * @var string $name
     *
     * @ORM\Column()
     */
    private $name;

    /**
     * @var integer $active
     *
     * @ORM\Column(type="boolean")
     */
    private $active = false;

    /**
     * @var string $description
     *
     * @ORM\Column(type="text", nullable=true)
     */
    private $description = null;
    ...

    /**
     * @param string $name
     */
    public function setName($name)
    {
        $this->name = $name;
    }

    /**
     * @return string
     */
    public function getName()
    {
        return $this->name;
    }

}
```

Auch wenn in Doctrine vieles formal vorgegeben wird - der Tabellennamen darf frei gewählt werden

Bei Doctrine‘s ORM kommen den Kommentaren ein einer Klasse eine tragende Rolle zu: durch diese werden Typ, Attrribute sowie Beziehungen festgelegt. Der Gedanke dahinter ist, dass auf diese Art keine zusätzlichen Konfigurationsdateien erstellt werden müssen, sondern alles „an einem Platz geschieht“. Das Codebeispiel zeigt ein einfaches Beispiel eines Productmodels (Auszug)

Zusätzlich zu den klassischen Settern und Gettern kann in den Models die Geschäftslogik implementiert werden, z.B. ein das Einrechnen von Betriebskonstanten eines Standorts oder fest hinterlegte Rabattkonditionen.

Abbildung „Doctrine - Definition Model - Auszug“ [51]

An weiteren Codebeispielen kann man beispielhaft sehen, wie Doctrine dafür verwendet werden kann, innerhalb einer PHP Klasse eine zugehörige Datenbanktabelle zu definieren, 1-N Beziehungen zu hinterlegen, Pagination zu verwenden und Daten aus mehrerern Models gleichzeitig anzuziehen. Letzteres erfordert zwar nicht mehr das „Joinen“ von Tabellen wie bei klassischem SQL, da die Beziehungen bereits über die Kommentare hinterlegt sind - dafür geschieht dies in der sogenannten **Doctrine Query Language**.
Diese setzt zwar die Konzepte von relationalen Datenbanken in Kombination mit Objektorientierung um, muss jedoch wiederum erlernt werden - was bei mir, der viele Jahre Erfahrung mit SQL hat und damit routiniert arbeiten kann, als klarer Nachteil erscheint.
Sicher: bewegt man sich erst einmal ausschliesslich mit objektoreintiertem PHP vorwärts, mag dieser Ansatz eine Erleichterung sein.
Der Vorteil von klassichen, relationalen Datenbanken, dass sich deren Strukturen seit vielen Jahren nicht grundsätzlich geändert haben, was einen routinierten Umgang mit ihnen ermöglicht, bleibt m.A. nach bestehen. Da Doctrine bei einem Model jedoch ohne „magic methods (__get() / __set()) arbeitet, also Getter und Setter für jedes Atrribut vorschreibt, können diese in einer IDE ausgelesen werden und stehen dort als Autocomplete beim Entwickeln zur Verfügung - ohne jedes Mal in der Tabellenstruktur der beteiligten Tabellen nachschauen müssen - dies ist positiv zu bewerten.


```php
class Greeting extends Doctrine_Record
{
    public function setTableDefinition()
    {
        $this->setTableName('greetings');
        $this->hasColumn(
            'content',
            'string',
            255,
            array(
                'type' => 'string',
                'length' => '255'
            )
        );
    }
}
```

Abbildung „Doctrine Codebeispiel Tabellendefinition“ [50]

```php
public function setUp()
{
    $this->hasMany(
        'Comments as Comments',
        array(
            'refClass' => 'Comment',
            'local' => 'id',
            'foreign' => 'user_id'
        )
    );
```

Abbildung „Doctrine Codebeispiel Beziehungen“ [49]

```php
<?php
use Doctrine\ORM\Tools\Pagination\Paginator;

$dql = "SELECT p, c FROM BlogPost p JOIN p.comments c";
$query = $entityManager->createQuery($dql)
                       ->setFirstResult(0)
                       ->setMaxResults(100);

$paginator = new Paginator($query, $fetchJoinCollection = true);

$c = count($paginator);
foreach ($paginator as $post) {
    echo $post->getHeadline() . "\n";
}
```

Abbildung „Doctrine - Codebeispiel Pagination“[52}

10.1.8 Das Backend

Das Backend ist für den Betrieb mit den Browser Chrome oder dessen Ablegern optimiert, da dieser das verwendete Javascript-Framework „extJS“ am performantesten umsetzt.
Es macht einen wertigen, einheitlichen Eindruck, benötigt jedoch bis zu 200 KB Arbeitspeicher. Die gesamte GUI wirkt „aus einem Guss“, sowohl von Design her wie auch bei der Anwendungslogik.
In den Abbildungen „Shopware - Backend - Übersicht 01“ und „Shopware - Backend - Übersicht 02“ habe ich die Grundstrukur des Backends sichtbar gemacht.

Backend

Klar zu erkennen ist die Einheitlichkeit im Aufbau - jeder Menüpunkt besitzt ein sprechendes Icon, was die Usability erhöht.
Gut gelöst sind die Suchfelder - das prominent sichtbare Suchfeld liefert (performant) Artikel bzw. Kunden (vgl. Abbildung „Shopware - Backend - Schnellsuche“), das Suchfeld bei den Grundeinstellungen (in Abb. „Shopware - Backend - Grundeinstellungen“ rot markiert) ist eine grosse Hilfe beim Auffinden von Einstellungen.
Allein dieses Feature spart eine Menge Zeit und Nerven.
Auf mich persönlich macht das Backend einen sehr durchdachten Eindruck, in das die Erfahrung von kompetenten E-Shop Entwicklern eingeflossen ist. Dieser Eindruck wurde kundenseitig bestätigt, was sowohl bei der Aquise wie auch bei der Markteinführung des fertigen Shops von Bedeutung war: versteht der Kunde intuitiv, wo und wie etwas zu bedienen ist, sinkt der Support-Aufwand auf Agenturseite und der Shopbetreiber kann effizient arbeiten.
Sämtliche Bereiche des Backends vorzustellen, würde den Rahmen dieser Arbeit sicherlich sprengen - daher möchte ich einige Punkte herausgreifen, die ich als erwähnenswert einstufe - auch vor dem Hintergrund der in Kapitel 8 erarbeiteten Aspekte.

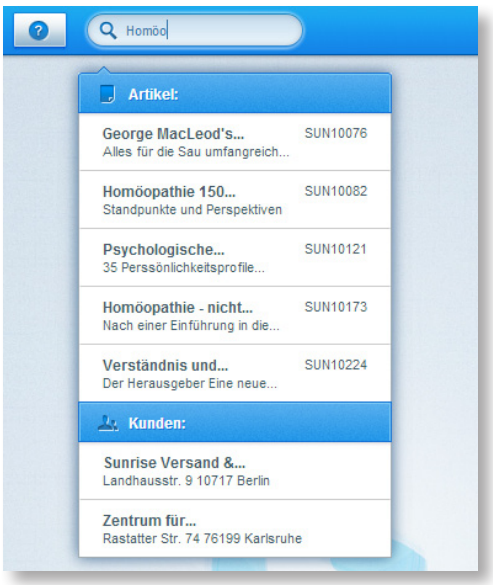


Abbildung „Shopware - Backend - Schnellsuche“ [55]

Die für das Backend verwendete Technologie „Ext JS“ von Sencha (http://www.sencha.com) ist eines der grossen GUI-Frameworks für Webanwendungen am Markt, mit Ausrichtung auf ein Komplettpaket im Bereich „Funktionsoberflächen“. Dieses werde ich im nächsten Kapitel vorstellen.

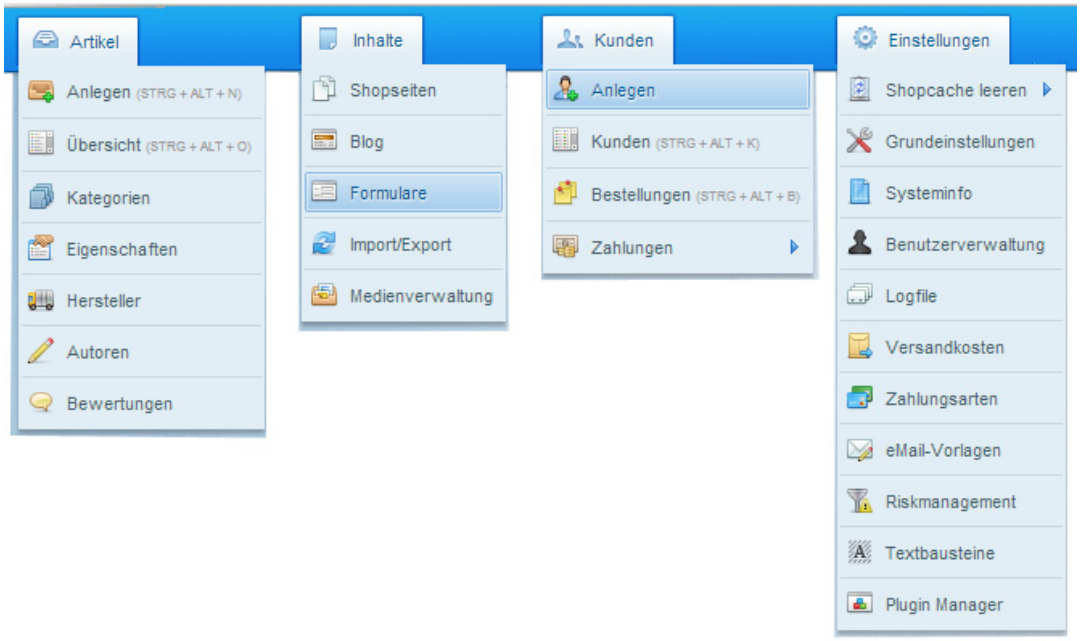


Abbildung „Shopware - Backend - Übersicht 01“ [53]

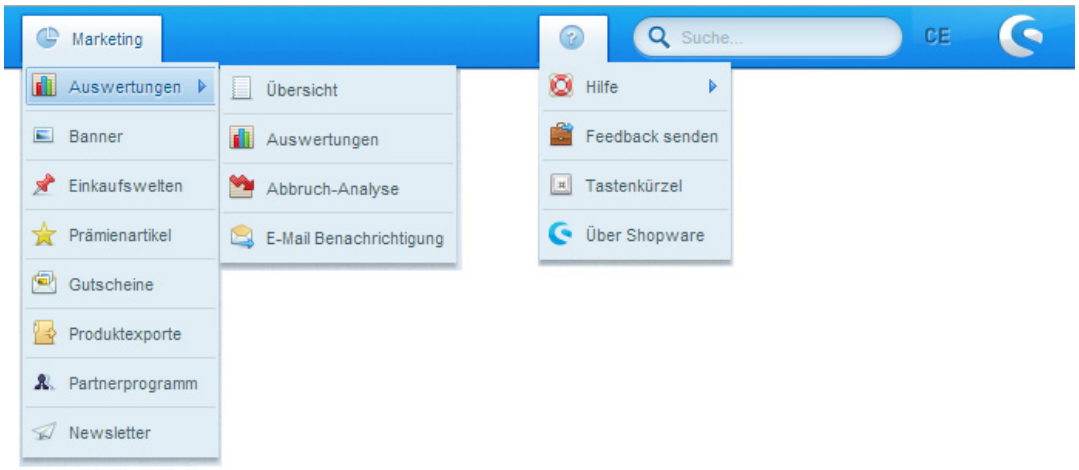


Abbildung „Shopware - Backend - Übersicht 02“ [54]

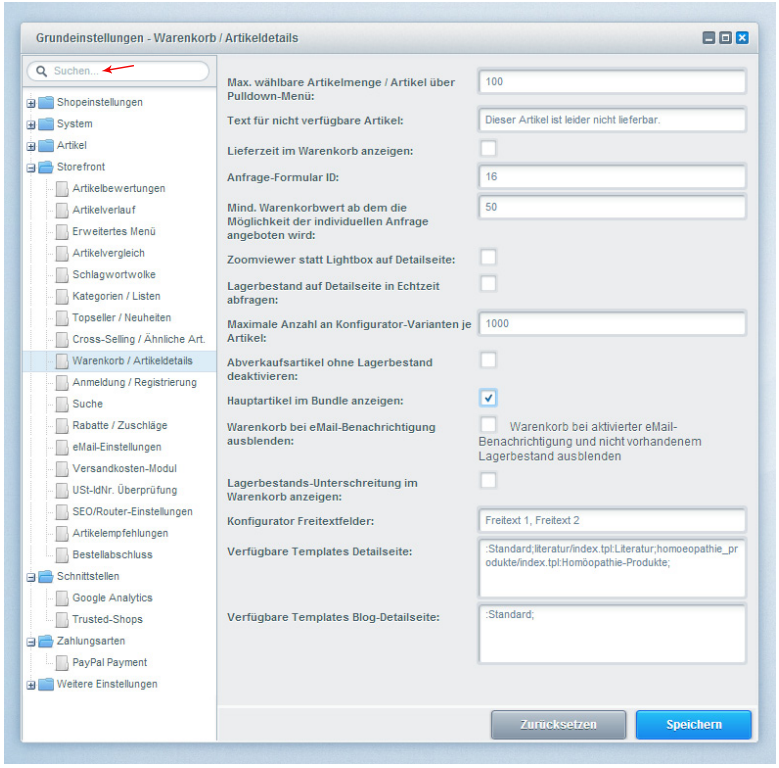


Abbildung „Shopware - Backend - Grundeinstellungen“ [56]

10.1.9 Die View - Sencha Ext Js und Smarty

Shopware verwendet für die Implementierung der View, also dem darstellenden Teil eines MVC Systems, zwei verschiedene Frameworks. Das Backend ist mittels Ext JS umgesetzt, beim Frontend kommt Smarty (http://www.smarty.net) zum Einsatz.

Smarty

Smarty ist eine Template-Engine und bietet die typischen Vorteile solcher Systeme: [Kompendium der Web-Programmierung]

Smarty

- Smarty ist sehr performant
- ein Template muss nur einmalig verarbeitet (compiliert) werden, wenn es sich nicht ändert
- Smarty bietet Kontrollstrukturen wie if/elseif/else/endif im Template
- Smarty unterstützt Caching für eine schnellere Ausgabe
- Smarty ist durch eine praktische Plugin-Architektur erweiterbar

Dabei kommt einem Templatesystem grundsätzlich diese Rolle zu:
Es nimmt die von Controller / Model ausgelieferten Daten entgegen und stellt diese mittels Template-Dateien dar.
Bei Smarty unterstützen diese das Prinzip der Vererbung, können also von anderen Template-Dateien erweitert werden.
Dabei organisiert sich Smarty in Blöcken, also namentlich ansprechbaren Darstellungseinheiten, die verschachtelt werden können, sowie eigentlichen Inhalt (HTML Code, Smarty-Includes, Variablen, die die vom Model gelieferten Inhalte beinhalten sowie Sprachvariablen)
Smarty unterstützt Funktionen wie Schleifen, If/else-Anweisungen oder Funktionen und bewegt sich daher in einer „Grauzone“, was die Konsequenz bei der MVC Implementierung betrifft. Zudem stehen eine Reihe Funktionen für die Modifikation des darzustellenden Inhalts zur Verfügung (klassische Stringfunktionen).

```
{extends file='parent:frontend/listing/index.tpl'}

{* Sidebar right *}
{block name='frontend_index_content_right'}{*$sCategoryContent|print_r*}{/block}

{* Tagcloud *}
{block name="frontend_listing_index_tagcloud"}
{if $sCloudShow}
        {action module=widgets controller=listing action=tag_cloud sCategory=$sCategoryContent.id}
{/if}
{/block}

{block name="frontend_listing_index_listing" append}
{/block}

{* Category text *}
{block name="frontend_listing_index_text"}
    {if !$hasEmotion && !$sSupplierInfo}
        {$smarty.block.parent}
    {/if}
{/block}
```

Ableiten und Erweitern des Templates

mittels PHP Funktionen können Inhalte ausgegeben werden (Debug)

Erweitern eines Blocks mittels „append“

if/else Anweisung

Abbildung „Smarty - Syntax Beispiel“

Der Ansatz, Template-Dateien nur bei Änderung des Inhalts neu zu kompilieren und ansonsten nur auszuliefern, überzeugt.
Die seitens Shopware eingerichteten Blöcke für die Darstellung der Frontendelemente sind vergleichbar zur Namenskonvention von Klassen beim Zend Framework benannt, so dass eine Orientierung beim Entwickeln leicht fällt.
So findet man den Block {block name="frontend_listing_index_banner"} im Ordner \frontent\listing\index.tpl.
Innerhalb von Shopware wird nur eine Teilmenge von Smarty's Funktionen eingesetzt, die wiederum herstellerseitig gut dokumentiert ist (dies macht Sinn, da man sich bei der offiziellen Smarty-Dokumenation schnell in der Funktionsvielfalt von Smarty verliert). Dies erlaubt grundsätzlich eine Beteiligung von Designern am Entwicklungsprozess.

Sprachvariablen

Auch die Integration von Sprachvariablen ist simpel, aber wirkungsvoll implementiert: so entspricht eine Variable vom Typ „s" jew. einem

```
{s name='ListingBoxLinkDetails'}{/s}
```

Textbaustein im Shopware Backend - diese sind mit Namespaces organisiert, so dass Sprachvariablen mit gleichen Namen unterschiedlichen Inhalt enthalten können.

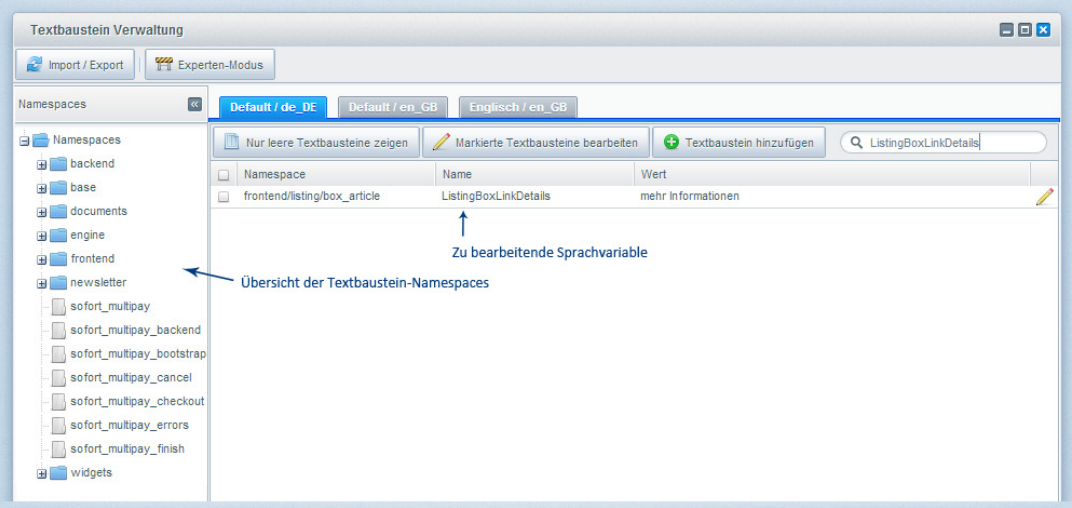


Abbildung „Shopware - Textbausteine" [61]

Einmal editiert, wird eine Sprachvariable von Smarty neu eingelesen und alle Templates, in denen diese vorkommt, werden neu kompiliert. Diese Architektur wirkt professionell und ist intuitiv zu bedienen.

Ext JS

Sencha‘s Ext JS kann man getrost zu den Platzhirschen der funktionalen View-Frameworks zählen. Konsequent weiterentwickelt (und gerade in Version 5 mit einem neuen Major Release erneuerten), deckt Ext Js eine Fülle von Funktionen ab, mit denen Standardsituationen strukturiert und einheitlich gelöst werden können.
Standardsituationen aus Ext JS Perspektive sind diese:

1. standardisierte Interaktion mit einem System
2. Layoutdefinitionen für Funktionsoberflächen
3. Tabellarische Darstellung von Daten

Ext JS

4. Darstellung von Baumstrukturen
5. Einrichten von intelligenten Formulare
6. Kombination der genannten Elemente mit Filterfunktionen, Tabs, Headern, Toolbars
7. „Zeichnen“ von Diagrammen

Auch wenn Ext Js bei Shopware als „View“-Teil des Backends verwendet wird, ist es intern wiederum mittels Model, View und Controller organsisiert und unterstützt sogar Namespaces.

Als grosser Vorteil von extJS mag die Einheitlichkeit gelten, die sowohl im Design wie auch bei der Bedienung erreicht wird.
Ein klarer Nachteil ist dessen mässige Performance, weshalb extJS in Shopware mit Symfony 2 Cache-Funktionen kombiniert wird.
Hinzu kommt die eher hoch anzusiedelnde Einarbeitungszeit, da das Konzept von Ext JS gewöhnungsbedürftig ist.
Sencha versucht dies damit zu kompensieren, dass herstellerseitig eine strukturierte, umfangreiche und ansprechend gestaltete Dokumentation mit vielen Beispielen zur Verfügung gestellt wird:

http://docs.sencha.com/

Ergänzt wird dies durch die Oberfläche „Sencha Try“, bei dem man sich auf der Grundlage von Beispielelementen, deren Erzeugungscode gleichzeitig angezeigt wird, einen ein Bild machen kann, wie Ext JS „tickt“. Diese Oberfläche erinnert ein wenig an das Javascript-Testportal „Js Fiddle“ (http://jsfiddle.net) und erlaubt ein Verändern / Ausführen des angezeigten Codes, so dass der Lerneffekt wirklich hoch ist:

http://try.sencha.com/extjs/4.2.0/

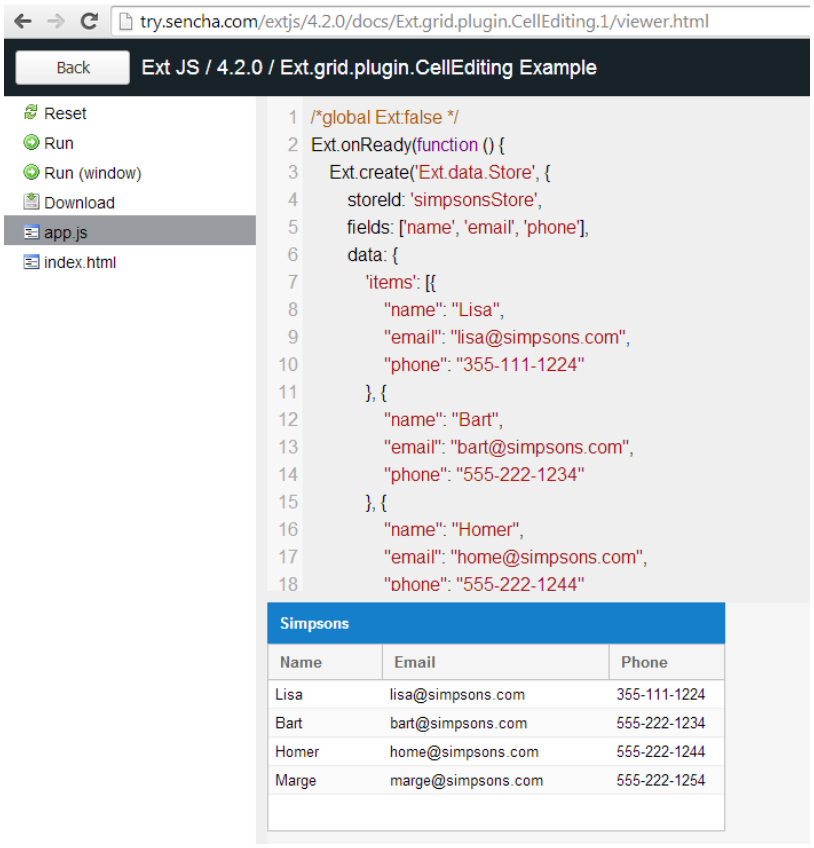


Abbildung „Ext JS - Try - Beispiel „Cell Editing“ [63]

http://try.sencha.com/extjs/4.2.0/docs/Ext.grid.plugin.CellEditing.1/viewer.html

Grundsätlich kann man festhalten, das Sencha das Thema Documentation und Community sehr ernst nimmt und hierfür Inhalte auf hohem Niveau liefert. Auch kann man anhand der Release-Zyklen sowie der allgemeinen Einschätzung des Herstellers, der sowohl mit Ext JS, wie auch den weiteren Produkten von Sencha

- Sencha Touch - Ausrichtung auf Mobilgeräte und Tablets
- Sencha GXT - Erstellen von HTML 5 Anwendungen auf Java-Basis
- Sencha Cmd - Entwicklungswerkzeug für Scaffolding von Skeleton-Applikationen und mehr

konsequent Innovation vorantreibt, von einem zukunftssicheren Framework sprechen.
Der Aufwand, den Ext JS bis zur sicheren Verwendung in einem Entwicklerteam einfordert, ist jedoch als hoch einzuschätzen.,
was - möchte man die Stärken von Ext JS nutzen und es als einziges Mittel zum Aufbau von Funktionsoberflächen einsetzen, eher bei einem langfristig ausgelegten Projekt Sinn macht, was auf Shopware zutrifft. Auch eignet es sich aufgrund der objektorientierten Struktur dazu, Elemente bei Bedarf zu erweitern, eine Möglichkeit, von der Shopware Gebrauch macht.

11. Shopware - Elemente und Konzepte

Nachdem nun alle verwendeten Frameworks vorgestellt sind, möchte ich darauf eingehen, wie diese innerhalb Shopware implementiert sind.

11.1 Shopware Controller

Shopware gliedert seine Controller in folgende Typen:

- Frontend Controller
- Backend Controller
- Widget Controller
- Api Controller

Shopware Elemente

Dabei kommen verschiedene Standard- und Helferfunktionen zum Einsatz - hier an einem einfachen Controllerbeispiel zu sehen.

```
class Shopware_Controllers_Frontend_Tutorial extends Enlight_Controller_Action
{
    public function init() {...}                  → Helferfunktion, wird vor dem preDispatch aufgerufen. Inhalt: Initialisierung
                                                    von Elementen, die beim Aufruf dieser Klasse immer zur Verfügung stehen soll.
    public function preDispatch() {...}           → Standardfunktion, kan für das Ausführen von Code vor dem Dispatch verwendet werden

    public function postDispatch() {...}          → Standardfunktion, kan für das Ausführen von Code nach dem Dispatch verwendet werden

    public function indexAction() {...}           → Standardfunktion, die aufgerufen wird, wenn keine andere Action im Request angegeben wird

    protected function getArticles() {...}        → beispielhafte, individuelle Controller Funktionen, die ja nach Action im Request
                                                    ausgeführt werden
    private function getArticleTranslation() {...}
}
```

Weitere Standardmethoden sind diese:

METHODS

- *Returns front controller*

```
Front()
```

- *Returns a class instance.*

```
Instance()
```

- *Returns request instance*

```
Request()
```

- *Returns response instance*

```
Response()
```

- *Returns view instance*

```
View()
```

Mit der Funktion „Front()“ kann auf den Enlight_Controller_Front zugegriffen werden, der für den Standard-Dispatch zuständig ist,die Funktion „Request()“ liefert das HTTP-Request-Objekt zurück, analog dazu erhält man via „Response()“ das HTTP-Response-Objekt, womit u.a. evtl. aufgetretene Exceptions ermittelt werden können.

Shopware Backend Controller leiten sich von der Klasse „Shopware_Controllers_Backend_ExtJs“ ab. Der Aufruf eines Backend Controllers leitet sich aus der jew. URL ab - so ruft die URL „www.myshopware.de/backend/ArticleList“ den Controller „ArticleList“ auf.

Erwähnenswert sind hierbei sind die Funktionen „LoadAction()“ (lädt die benötigten Javascript-Dateien für Ext JS Erweiterungen) sowie „ExtendsAction()“ (kann zur Ableitung von bestehenden Backend-Modulen verwendet werden).

11.2 Shopware Events und Hooks

Events und Hooks sind beides Werkzeuge, mittels denen ein Shopware-System angepasst werden kann. Üblicherweise wird dies via Plugin organisiert. Shopware unterscheidet zwischen klassischen Events, also Ereignissen, die beim Aufruf von Standardfunktionen ausgelöst werden und an die man sich „anhängen“ kann, sowie den sog. „Hooks“, mit denen das System zur Laufzeit weitestgehend frei modifiziert werden kann.

Events

Die Controller Action Funktionen lassen sich via Event abfangen und verändern, wobei folgende Syntax eingehalten werden muss:

Shopware Elemente

Enlight_Controller_Action_MODULE_CONTROLLER_ACTION

MODULE = welcher Bereich, also Frontend oder Backend

CONTROLLER = welcher Controller, in diesem Fall der Checkout-Controller

ACTION = an welche Action des Controllers möchte man sich anhängen ?.
die eigentliche Controllermethode lautet dabei
[NAME_DER_ACTION]Action().

Eine konkrete Eventregistrierung für den Standard-Checkoutprozess geschieht auf diesem Weg:

```
$this->subscribeEvent(
   ‚Enlight_Controller_Action_Frontend_Checkout_Confirm‘,
   ‚onCheckoutConfirmAction‘
);
```

Der zugehörige Listener sieht dann so aus:

```
public function onCheckoutConfirmAction(Enlight_Event_EventArgs $arguments)
{
   /**@var $checkoutController Shopware_Controllers_Frontend_Checkout*/
   $checkoutController = $arguments->getSubject();
   $request = $checkoutController->Request();
   $response = $checkoutController->Response();
}
```

Auf diese Art kann die originale Methode „confirmAction()“ mit eigenem Code überschrieben werden.

Hooks

Da nicht jede Methode in Shopware Events aufweist, was einer Beschränkung bei der Modifikation des Systems mittels Events darstellt, bietet Shopware zusätzlich sog. „Hooks“ an, die einen eigenen Schicht

im Pluginsystem darstellen. Umgesetzt wurde dies mit dem vorhin erwähnten Proxie-Entwurfsmuster: dieses erlaubt eine Modifikation von Instanzen bestehender Klassen zur Laufzeit. Dies bedeutet konkret, dass Funktionalitäten einer Klasse verändert werden können, ohne in der eigentlicher Klasse etwas ändern zu müssen. Ein „manuelles Extenden“ der Klasse ist nicht nötig.

An einem Bespiel wird dies deutlich:
Möchte man einen Hook für die Function „sGetArticleById()“ in der Klasse „sArticles“ registrieren, die das Verhalten der Funktion ändert, so wird dies so durchgeführt (Hooks werden wie Events mittels der Helferfunktion „subscribeEvent() registriert):


```
$this->subscribeEvent(
   ‚sArticles::sGetArticleById::before‘,
    ‚beforeGetArticleById‘
);
```

Dies entspricht diesem Raster:

```
$this->subscribeEvent(
   ‚CLASS::FUNCTION::TYPE‘,
   ‚HOOK-LISTENER‘
);
```

Der zugehörige Listener darf natürlich nicht vergessen werden:

```
public function beforeGetArticleById(Enlight_Hook_HookArgs $arguments)
{
   $articleId = $arguments->getId();
   $categoryId = $arguments->get(‚sCategoryID‘);
   $sArticle = $arguments->getSubject();// Original-Klasse, von welcher
                                        //der Hook abstammt
    $arguments->set(‚id‘, 3);//Setzen der CategoryID
}
```

Ist es notwendig, Shopware Funktionen mitttels Hook vollständig durch eigene Methoden zu ersetzen, so ist dies per sog. „Replace Hook“ ebenfalls möglich:

```
$this->subscribeEvent(
   ‚sArticles::sGetArticleById::replace‘,
   ‚replaceGetArticleById‘
);
```

Der zugehörige Listener

```
replaceGetArticleById(Enlight_Hook_HookArgs $arguments){}
```

überschreibt in diesem Fall die Standarfunktion „sGetArticleById()“ und übernimmt die Rückgabe dieser Funktion per

arguments->setReturn($articleData); .

Sollte dies noch nicht ausreichend sein, obliegt dem Entwickler zus. die Option, eigene „hoookable“ Klassen ins System einzuführen - dies kann dies mit dem Enlight_Hook Interface bewerkstelligt werden.

Der Mehrwert der Hook-Architektur ist deutlich zu erkennen:

- ohne Beschränkung auf das Vorhandensein von Methoden, an die Events gebunden können, kann das System erweitert werden
- Das Verhalten von Klassen kann je nach Bedarf verändert werden, ohne die Dateien des Kernsystem
- Das Kernsystem bleibt „unangetastet“ funktionsfähig, da Hooks zwar in das Verhalten des Systems eingreifen, nicht jedoch dessen physikalische Dateien modifiziert.

Dies klingt in der Theorie einfach und logisch - in der Praxis jedoch wird es spätestens dann kritisch, wenn Hooks nicht Shop-Inhalte auf eine andere Art ausliefern, sondern aktiv in die Geschäftslogik eingreifen. Dies kann Situationen schaffen, die nicht mittels „Plugin deaktivieren“ wieder rückgängig gemacht werden können.
Auch ist es ohne grössere Erfahrung nicht leicht einzuschätzen, welche Seiteneffekte durch eine Verhaltensänderung von Teilen des Systems im Gesamtsystems ausgelöst werden (Dateninkonsistenz, Instabilität, Performanceprobleme, harte Fehler).

11.3 Shopware Models

Als Model kommt bei Shopware das vorhin beschriebene Doctrine ORM zum Einsatz. Zur Erinnerung: ein Model ist bei Doctrine eine Kombination aus einer PHP Klasse sowie der zugehörigen Datenbanktabelle. Als Vorteile dieser Architektur nennt die Shopware Dokumentation [TODO] diese Aspekte:

1. *Möglichkeit zur zentralen Definition der Datenbankstruktur in PHP*
2. *Verwaltung aller Queries (also Datenbankabfragen) in einem zentralen Repository*

Umgesetzt wurden dies beispielsweise mittels des Validator\Constraint-Pakets vom Symfony. Dies beinhaltet Funktionen für die Validierung von Daten gegenüber sog. Constraints, wobei eine Reihe gängiger Prüfmuster für verschiedene Anwendungsbereiche (Email, Kreditkar-

tennummer, Ländercodes und vieles mehr) genutzt werden können [42].

Weitere häufig verwendete Elemente ist die ArrayCollection von Doctrine,welches eine Art übergeordnete Sammlung von PHP Arrays erlaubt. Grundlage eines Models ist die seitens Shopware auf Doctrine-Basis entwickelte „Model Entity“, die der Verwaltung der Standard-Models dient und als abstrakte Klasse den Bauplan für die Funktionen liefert, die die Models enthalten können / müssen.
Das Mapping-Paket von Doctrine kommt „out-of-the-Box“ zum Einsatz.

Shopware Elemente

```
use Symfony\Component\Validator\Constraints as Assert,
    Doctrine\Common\Collections\ArrayCollection,
    Shopware\Components\Model\ModelEntity,
    Doctrine\ORM\Mapping AS ORM;
```

Shopware organisiert sein System auf Datenhaltungsebene nahezu lückenlos mittels Models, was den Umgang mit diesem Aspekt vereinheitlicht und übersichtlich macht. Eingesetzt werden die klassischen Elemente von Doctrine:

- Column Annotations (= Definieren einer Klassenvariable als Datenbankspalte per `@ORM\Column`)
- Validierungs Annotations (Validierung von Daten mittels eigener oder vorgefertiger Constraints)
- Assoziationen (OneToOne / OneToMany / ManyToOne / ManyToMany)
- Cascadierung (Weitergabe von Aktionen auf einem Model an verknüpfte Models)
- Model Events (Insert / Update / Remove)
  Da Doctrine von Haus aus Events unterstützt, wurden diese in das Shopware-Eventsystem integriert. Für jede der 3 genannten Operationstypen wird jeweils ein „pre-“ und ein „post-“Event ausgelöst, welches von Plugins ausgewertet werden kann.

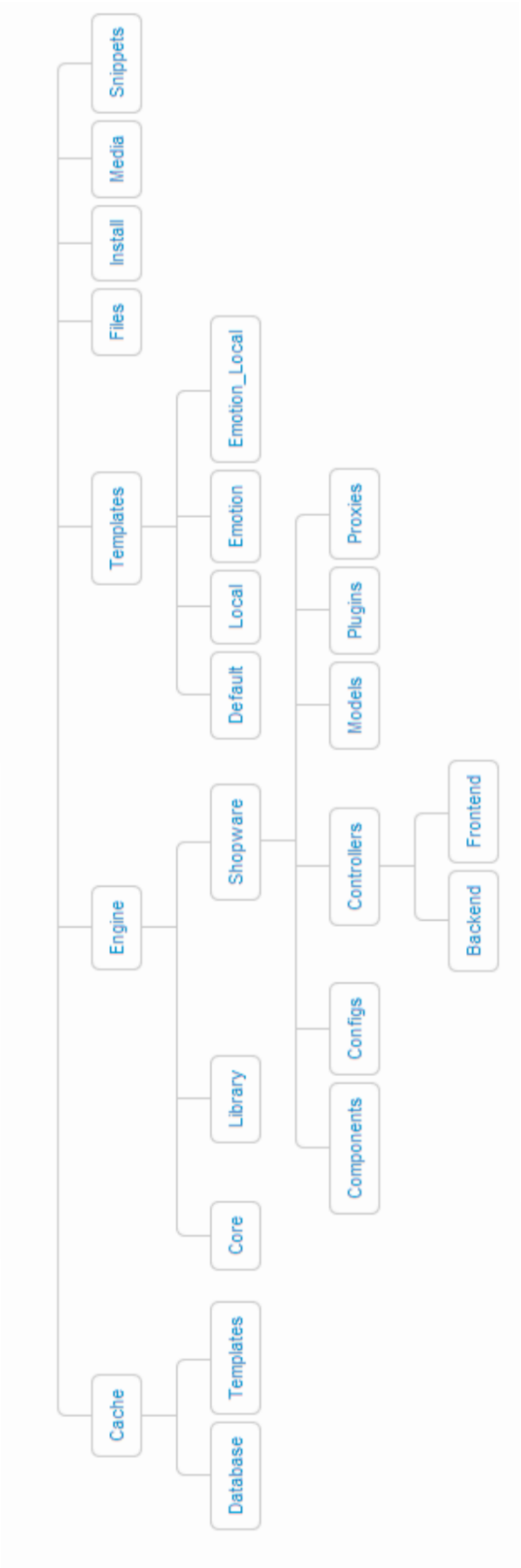


Abbildung „Shopware Ordnerstruktur“ [72]

12. Implementierung: www.sunrise-versand.de

Im Kapitel „Imlementierung“ möchte ich sowohl auf die Umsetzung des „eigentlichen“ Shop eingehen und anhand ausgewählter Anwendungsbeispiele die Analyse von Shopware anschaulich fortsetzen.
Dabei habe ich mich auf einige ausgewählte Aspekte beschränkt, da meine pratische Arbeit im Sommersemester 2014 die exemplarische Programmierung eines Backend-Plugins beinhaltete, auf die ich detailliert eingehen möchte.

12.1. Verwendete IDEs

Für die Implementierung des Online-Shops habe ich auf Code-Ebene mit der IDE „Netbeans“ in Version 7.2 gearbeitet.

Projektwerk-
zeuge

Diese IDE ist schlank gehalten und verfügt dennoch über die meisten Features, die für ein Projekt dieses Typs benötigt werden.

Hervorzuheben ist dabei:

- nativer PHP Support mit Code-Highlighting und Autocomplete
- Auffinden von Klassen- und Variablenverwendung für Code-Analyse
- Möglichkeiten für Debugging (Breakpoints, Debug-Sessions)

Für Datenbankoperationen kam die rein für Datenbankzwecke spezialisierte IDE „Aqua Data Studio“ zum Einsatz, die Key-Features wie Autocomplete sowie Generierung von Standardoperationen per Kontextmenü bietet. Ausserdem konnte ich auf diese Art in meiner gewohnten Arbeitsumgebung arbeiten - die browserbasierende-Oberfläche von mySQL empfinde ich als wenig komfortabel.

Bei der Plugin-Programmierung habe ich mit der IDE „PHP Storm 7“ gearbeitet, zum einen, da Shopware diese IDE intern verwendet und z.T. in Herstellertutorials darauf Bezug nimmt, zum anderen, da PHP Storm eine extrem stabile, schnelle und leistungsstarke IDE ist.
Die Stärke dieser IDE wird deutlich, wenn man mit einer objektorientierten Applikation zu tun hat die - lokal installiert - in der IDE per Code-Autocomplete zur Verfügung steht, wie es z.B. bei einer Java-Applikation der Fall ist.

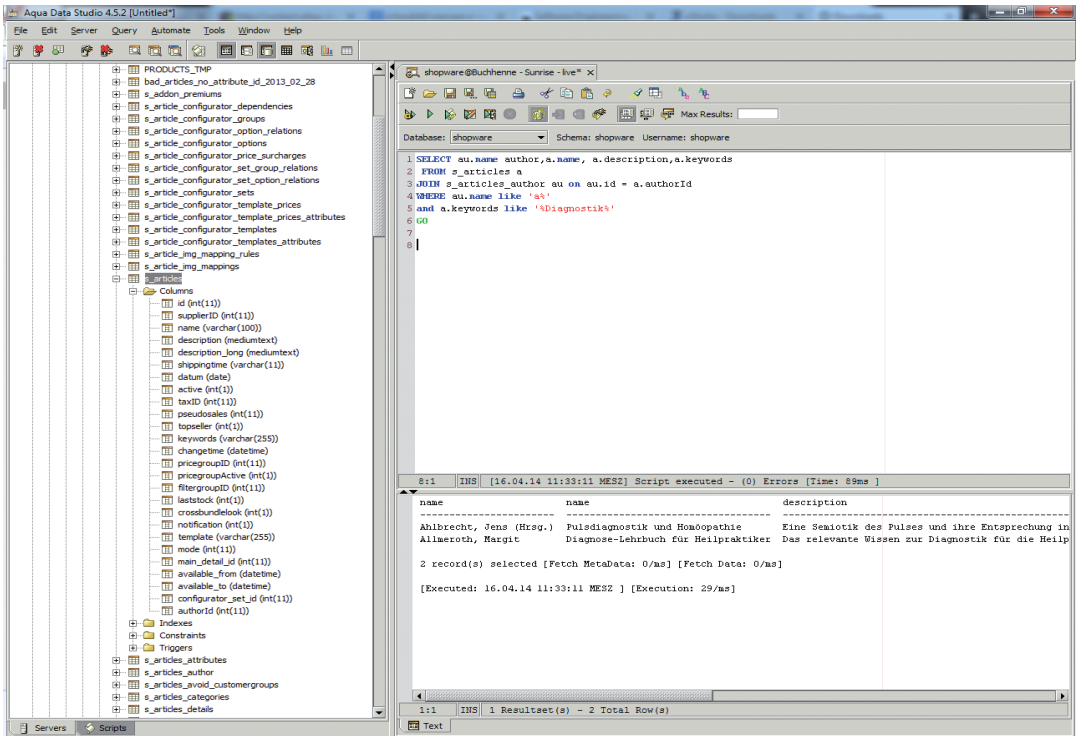


Abbildung „Aqua Data Studio - IDE für Datenbankoperationen

Projektwerk-
zeuge

Dies ist auch bei Code-Tracing, sprich Analysieren der Code-Strukturen durch Anzeige von Dingen wie „wo ist diese Klasse deklariert“, „wo wird diese Variable verwendet“, von grossem Vorteil.

Zum Vergleichen von Ordnern und Dateien habe ich unter Windows mit Araxis Merge 6.5 sowie unter Ubuntu mit Meld 1.8.6 gearbeitet.

12.2. Coding Standards

Hinsichtlich der Coding Standards für die Shopware Pluginentwicklung macht der Hersteller konkrete Vorgaben, an denen ich mich orientiert habe. Dabei wird ein einheitlicher Stil für die generelle Vorgehensweise, Naming und Statement Conventions vorgegeben: http://wiki.shopware.de/Shopware-Coding-Standards_detail_661.html

12.3. Einrichten des Projekts

Dem Einrichten des Projekts kommt eine deutlich grössere Bedeutung zu, als ich am zu Beginn wahrhaben wollte. Konnte ich innerhalb der Frontendentwicklung im Rahmen des Hauptprojekts, dem Relaunch noch gut mit der Kombination FirePHP / Shopware-Debug-Plugin debuggen, so war dies bei der Entwicklung des Backend-Plugins zuverlässig nur mit der lokalen Deployment-Einrichtung des (Ubuntu mit nativem Apache) Projekts in Kombination mit Xdebug zu bewerkstelligen. Dies habe ich konsequent mit PHP Storm 7 umgesetzt und mich dabei an die Herstelleranleitung gehalten: http://wiki.shopware.de/_detail_1067_869.html

Seitens Equinoxe wurden mir - für den Relaunch - ein virtueller Debian-Webserver als Entwicklungsserver bereitgestellt, bei der Pluginentwicklung diente mir der hauseigene Shopware-Entwicklungsserver als Testumgebung.

Die URLS der Entwicklungsserver sind:

http://sunrise.entwicklungsserver.de
http://shopware.entwicklungsserver.de

12.4. Debugging

Debugging

Beim Umsetzung des Relaunchs sowie der Programmierung des Backend-Plugins kamen folgende Debug-Werkzeuge zum Einsatz:

- Firebug für Firefox
- Fire-PHP in Kombination mit dem Shopware Debug-Plugin
- Das freie Plugin „Entwicklertoolbar für Shopware“
- XDebug

Firebug ist der Klassiker beim Debuggen im Webbereich und braucht m.A nach nicht gesondert vorgestellt werden. Fire-PHP setzt auf Firebug auf und erlaubt Debug-Ausgabe von PHP-Elementen, ohne mit var_dump / print_r arbeiten zu müssen, was einen „unbemerkten“ Debug ermöglicht, also auch für die Wartung eines Shops von Vorteil ist.

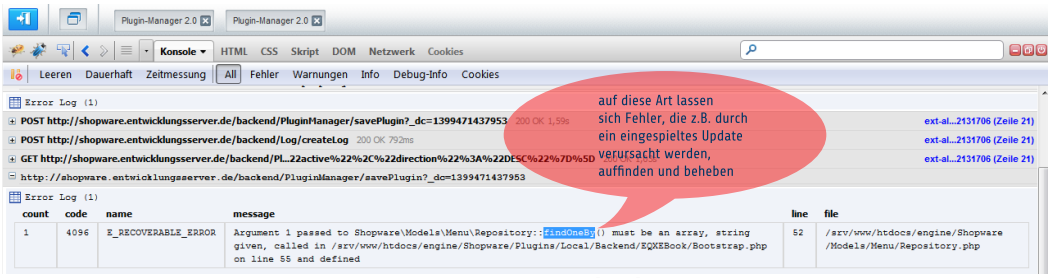
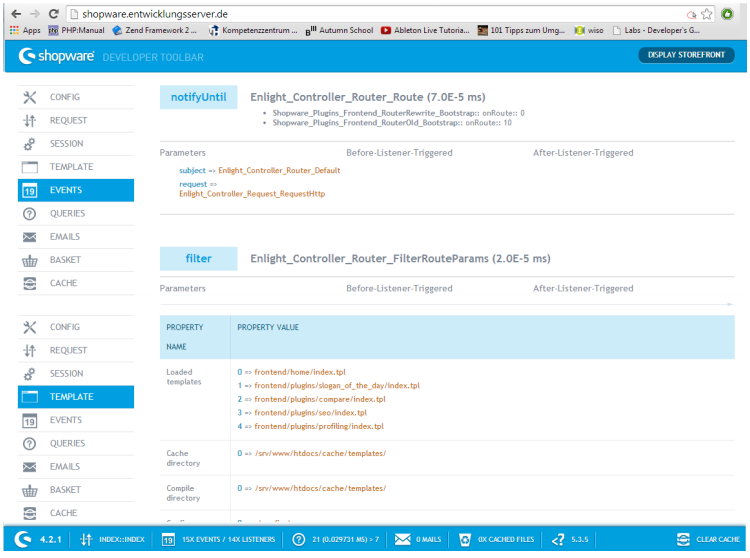


Abbildung „Shopware -. Debug mit Fire-PHP“ [65]



Die Entwicklertoolbar, enstanden im Rahmen eines der Shopware-„Hackathons“, bietet eine übersichtliche Darstellung relevanter Daten eines Requests (hier beispielhaft zusammengestellt: Events und Template).

Abbildung „Shopware -. Plugin Entwicklertoolbar“ [64]

12.5. Konzeption des Artikelsortiments

Die Konzeption des Artikelsortiments war eine der grössten und wichtigsten Aufgaben innerhalb des Projekts.
Dabei galt es, vor allem zwei Herausforderungen zu meistern - zum einen, die verschachtelte Kategoriestruktur zu straffen und zum anderen, einen Teil der Kategorien in Produktfilter zu wandeln.

Der alte Shop kann unter der URL **https://alt.sunrise-versand.de** eingesehen werden, damit die frühere Struktur greifbar wird.

Zusätzlich habe ich mich bei der Neukonzeption der Kategorien an der sogenannten 7 +/- 2-Regel [6] orientiert, die eine Aussage über die Kapazität des menschlichen Kurzzeitgedächtnisses macht:
Bis ca. 7 Elemente einer Navigation kann ein Mensch gut und schnell gleichzeitig erfassen.

Zusätzlich galt es, Kategorien, die inhaltlich beliebig zuordnungsfähig sind, im alten Shop jedoch als feste Kategorien geführt wurden, in filterbare Produkteigenschaften umzuwandeln.

Betrachten wir dies an einem Beispiel:

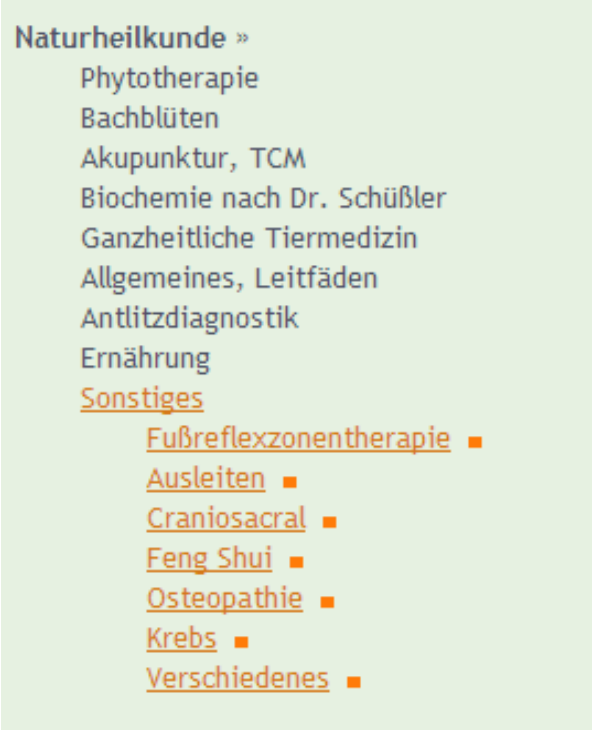


Abbildung „Sunrise Shop alt – dreistufige, steile Kategoriestruktur“[16]

Im alten Shop wurden Begriffe wie „Krebs“ als feste Kategorien geführt, was keine Flexibilität bei der Zuordnung von Artikeln zuließ. Spielte der Begriff „Krebs“ in einem Werk mit Hauptfokus auf Bachblüten eine Rolle, so ließ sich das nicht abbilden, da „Krebs“ bereits als Unterkategorie von Naturheilkunde ->Sonstiges vergeben war.

Daher konzipierte ich ein neues Kategorie- und Produktfiltersystem, welches in Kombination eingesetzt wird.
Shopware unterstützt sowohl auf Kategorie- als auch auf Produktfilte-rebene eine Mehrfachzuordnung, was die Flexibilität der Sortiments-struktur steigert.

Vergleichen wir nun die Hauptkategorien des alten Shops mit denen im Relaunch :

RADAR Preisaktion
Spitzentitel des Jahres
Sunrise_Buecherregal
Neuerscheinungen
Vorschau
Sonderpreise
Hahnemann
Materia Medica
Repertorien
Theorie
Praxis
Einsteiger und Laien
Verwandte Gebiete
Naturheilkunde
Medizin
Patientenkarteimappen / Fragebögen
RadarOpus Homöopathie-Software
Zeitschriften
Kassetten, CDs, Videos
Taschenapotheken
Displays und Sortiersysteme!
Homöopathie-Produkte
Englische Literatur

Abbildung „Hauptkategorien alter Shop“ [17]

Bücher	Quellen	Praxiszubehör	Software / Multimedia	Zeitschriften
Materia Medica	C.M. Boger	Lederetuis	Radar	AHZ
Repertorien	C.v. Bönninghausen	Sunrise-Lederetuis	Audio-CDs	ZKH
Theorie	S. Hahnemann	Sunrise-Stoffrollen	Video-DVDs	Homöopathie Konkret
Praxis	C. Hering	Homö-Set Lederetuis		Homöopathie Zeitschrift
Psychologie	G.H.G. Jahr	Weitere Etuis		Spektrum der Homöopathie
Einführung	J.T. Kent	Arzneibehältnisse		Homoeopathia Viva
Medizin	M. Mangialavori	Patientenkarteimappen		Tierhomöopathie
Weitere Therapien	R. Sankaran	Weiteres Zubehör		Ganzheitliche Tierhomöopathie
Vorschau	J. Scholten			DHZ
Neuerscheinungen	G. Vithoulkas			
Sonderpreise				
Bücher (engl.)				
Gutscheine				

Abbildung „Hauptkategorien neuer Shop“ [18]

Deutlich erkennbar ist, daß die ehemaligen 23 Hauptkategorien fünf Hauptkategorien gewichen sind.
Die rot markieren Unterkategorien bei „Bücher“ (gleiches gilt für die Zahl der im System geführten Quellen) entsprechen nicht dem 7 +/- 2 Paradigma, wurden jedoch vom Kunden explizit gewünscht.

Die Ziele bei der Strukturierung des Artikelsortiments in der Übersicht :

- Straffung der Kategoriestruktur
- Aufbau einer flachen, zweistufigen Kategoriehierarchie
- Berücksichtigung der 7 +/- 2-Regel
- Umwandlung von Kategorien in Produktfilter, sofern die Kategorie Merkmalcharakter hat

12.6. Konzeption der Produktfilter

Viele der vorherig als Kategorie geführten Begriffe wurden in Produktfilter überführt.

Bei der Konzeption wurden in Absprache mit dem Kunden Produktfilter für Literatur und Non-Books definiert.

Die Anzahl der im System angelegten Filter relativiert sich durch die Tatsache, dass im Frontend nur die Filterausprägungen angezeigt werden, für die in der aktiven Kategorie auch Produkte gefunden werden.

Literatur	Praxis	Therapie (Allgemein)
		Therapie (Kopf -> Fuß)
		Ratgeber
		Kinder
		Frauen
		Psychologie
		Onkologie
		Tiere / Pflanzen
		Fallsammlungen
		Medizin
	Sonderpreise	Vorschau
		Neuerscheinungen
		Preisaktionen
		Antiquariat
	Sunrise Bücherregal	Einsteiger / Laien
		Studium / Ausbildung
		Therapeuten
	Weitere Therapien	Heilpraktiker
		Medizin
		Ernährung
		Phytotherapie
		Biochemie (Schüßler)
		Geist und Seele
		Bach-Blüten
		TCM / Akupunktur
		Impfung
	Theorie	Lehrbücher
		Miasmen
		Anamnese
		Leitsymptome
		Handbücher (AML)
		Vorlesungen (AML)
		Arzneiprüfungen
		Einzelne Arzneimittel
		Lernhilfen
		Geschichte / Forschung
		Quellen
		Biografien
		Romane
		Ausbildung (HP)
		Prüfung (HP)
Non-Books	Sunrise Produkte	
	Arzneibehältnisse	
	Arzneimittelbriefchen	
	Alu-Etuis	
	Lederetuis	
	Patientenkarteimappen	
	Sortiersysteme	
	Büsten / Poster	
	Mörser	
	Rund ums Buch	
	Multimedia	
	Software	

Abbildung „Konzeption der Produktfilter“[19]

Die User Experience des Shop-Nutzers sollte möglichst hoch sein - was ich (bei einem Artikelsortiment von über 2500 Produkten) bei der Konzeption unter anderem dadurch realisierte, dass die Produktauswahl bei Navigation auf Kategorieebene mit Produktfiltern eingeschränkt werden kann, bis die gewünschten Artikel vorliegen.
Die zweite „grosse“ Möglichkeit beim Auffinden von Produkten istdie leistungsfähige Suchfunktion - ich werde gleich einen konkreten Use-Case durchspielen..

12.7. Autorenverwaltung

Die Autorenverwaltung stellte einen grösseren Eingriff ins System dar, der jedoch von zentraler Notwendigkeit für den Kunden war, da sich ein E-Shop für Literatur kaum ohne Autor abbilden lässt.
Die Implementierung der Autorenverwaltung wurde jedoch nicht von mir vorgenommen, so dass ich auf diese nur kurz eingehen möchte.
Umgesetzt wurde die Autorenverwaltung als funktionale Kopie der Herstellerverwaltung, mit allen benötigten Elementen wie der Anpassung des Ext-Js Teils des Backends, der benötigten Datenbanktabellen, der Erweiterung aller Artikel-spezifischen PHP Klassen sowie allen Template Dateien, bei denen Artikel eine Rolle spielen - was bei bei fast allen Template-Dateien der Fall ist.

Frontend

12.8. Frontend / Layout / User Experience

Bei der Umsetzung des Frontends wurde nahezu ausschliesslich mit Elementen gearbeitet, die das Standardtemplate von Haus aus unterstützt.
Dabei konnten alle Kundenwünsche umgesetzt werden:
Das Einrichten der Startseite als klassiche Übersichtseite mit Banner sowie grafischer Navigation für die Fallrätsel (ein Kundenbindungsinstrument von Sunrise) und die Seminare sowie die Anzeige von Neuerscheinungen (wahlweise vom System automatisch ausgewählt oder vom Redakteur so eingestellt).
Der Einstieg in eine Hauptkategorie mittels einer grafischen Navigation konnte mit den sog. „Einkaufswelten“ (eingeführt mit Shopware 4) mühelos umgesetzt werden - dabei kann via Ext Js basierendem WYSIWYG-Editor ein Raster aufgezogen werden, in welchem Elemente wie Banner, Kategorie-Teaser, unterschiedliche Slider, Youtube-Videos etc. platziert werden können.
.

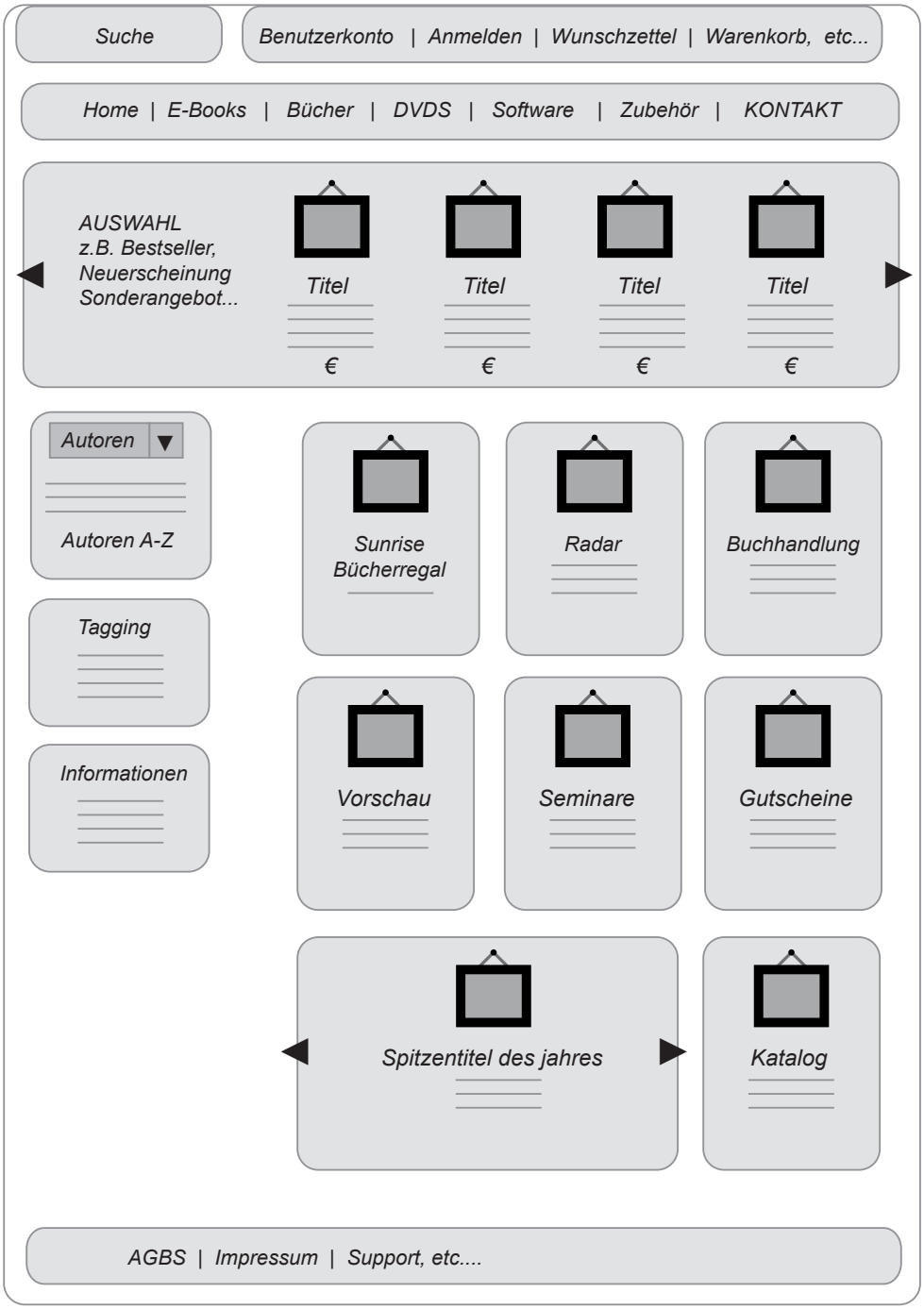


Abbildung „Frontend - Layout - Mockup Startseite“ [66]

Frontend

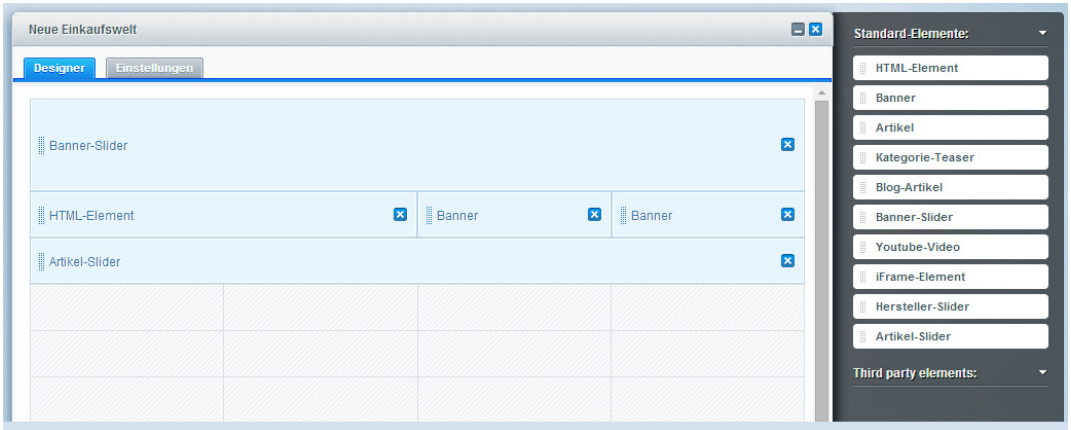


Abbildung „Shopware Einkaufwelten“ [68]

Frontend

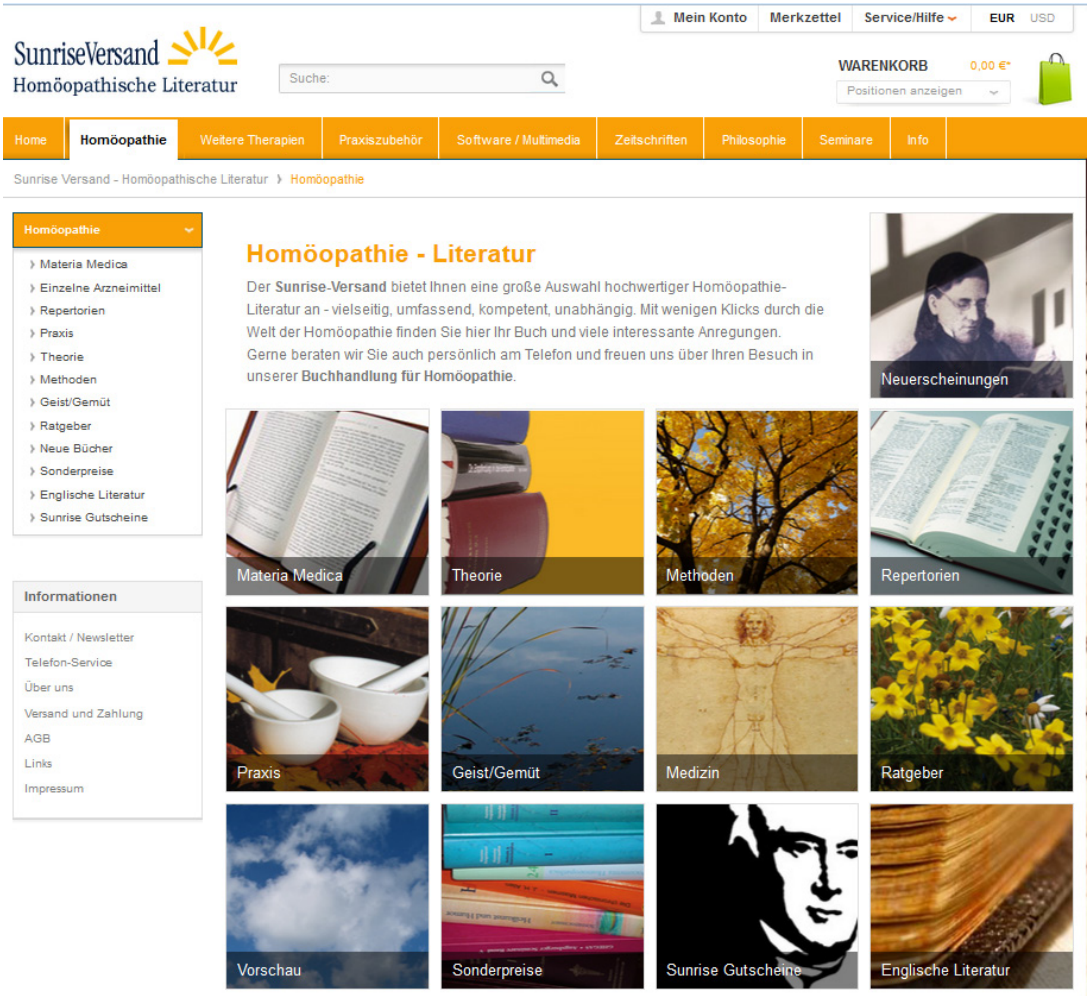


Abbildung „Frontend - grafische Kategorieansicht“ [67]

Elemente wie Up- oder Crosselling können im Backend nach Wunsch aktiviert werden - auch hier hat der Shopbetreiber die Wahl, ob das System dies automatisch auf der Grundlage von Auswahlalgorithmen bereitstellt, alternativ werden „handverlesene“ Artikel gezeigt.

Oberhalb der Artikelliste werden Produktempfehlungen von Sunrise für die jeweils aktive Kategorie angeboten - auch dies mühelos mit den Shopware-Standardfunktionen umsetzbar:

Abbildung „Frontend - Hauptkategorie Produktempfehlungen“[69]

Im folgenden möchte ich anhand eines Use-Cases deutlich machen, wieviele Einstiegsmöglichkeiten im Shop möglich sind, wie man durch Filterfunktionen zum gewünschten Produkt findet und wie stringent die Navigation umgesetzt ist.
Der User hat eine Hauptkategorie gewählt, was sowohl unmittelbar das Blättern in der Artikeliste erlaubt, in der linken Navigation jedoch eine automatisiert erstellte Auswahl an Filtern generiert,

Frontend

Abbildung „Frontend - Use Case Materia Medica 01“ [70]

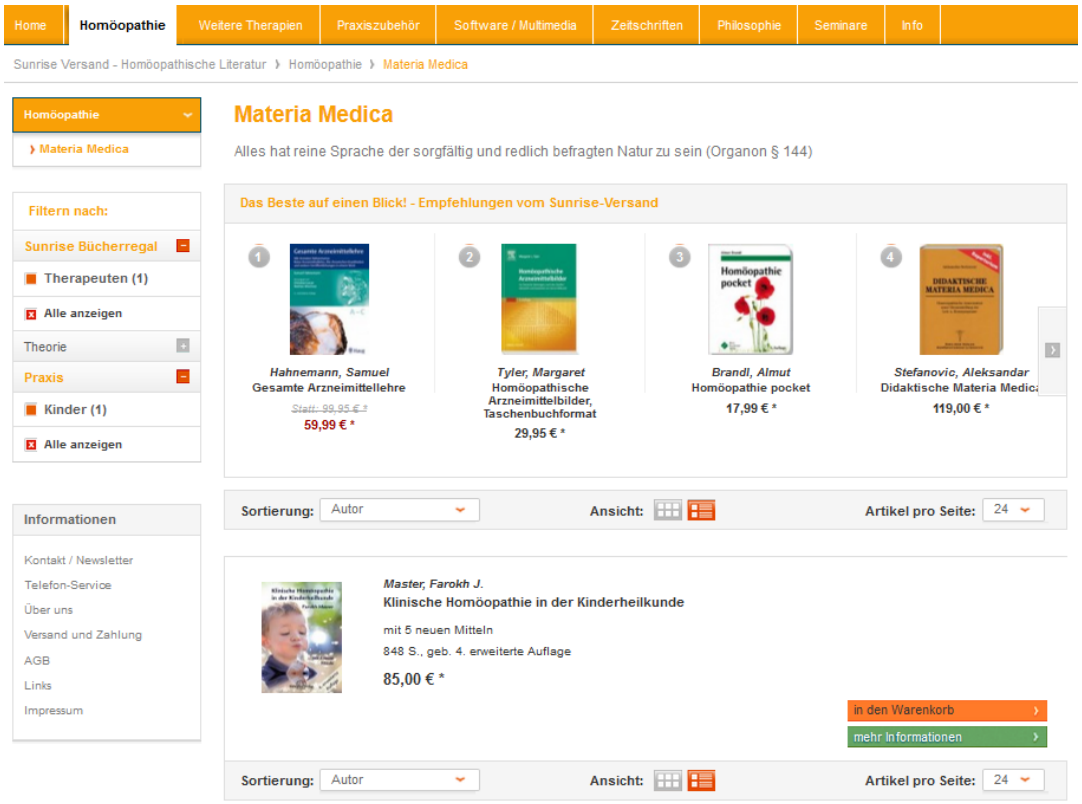


Abbildung „Frontend - Use Case Materia Medica 02“ [71]

die jeweils dann angezeigt werden, wenn es in der gewählten Hauptkategorie mindestens ein Produkt mit einer Eigenschaft der jeweiligen Filtergruppe gibt. Diese können kombiniert werden. So kann der User über die Filtergruppe „Sunrise Bücherregal“ eine Auswahl treffen, in welchem Teil des virtuellen Buchladens er stöbert: Laien, Ausbildung oder Therapeuten (im vorliegenden Fall: Therapeuten).

Bei „Praxis“ wählt er, welches der konkrete Anwendungsbereich, an dem er interessiert ist: in diesem Fall „Kinder“.
Mit nur 4 Mausklicks hat der User die Schnittmenge gebildet, an der er interessiert ist und kann in dieser blättern.

Intelligente Produktsuche

Ein weiterer, typischer Einstieg ist die prominent sichtbare Suche. Diese ist fehlertolerant und verarbeitet die Eingabe von Buchnamen oder Autoren sowie die Eingabe mehrerer Keywords wie „Therapeut Kinder“, wie im folgenden Fallbeispiel.

intelligente Suche

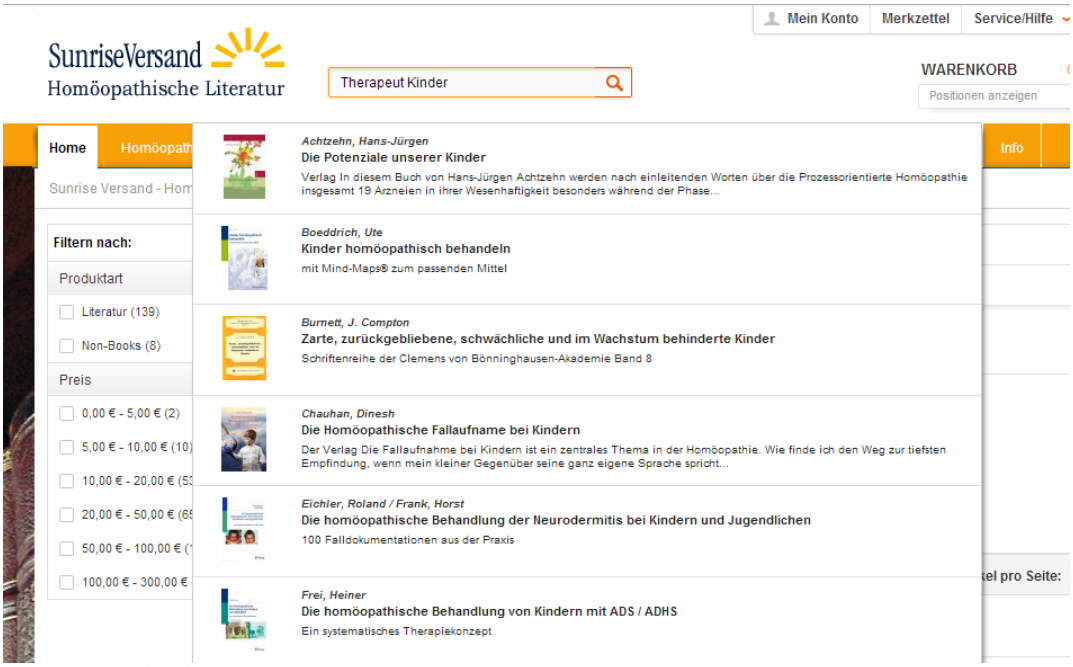


Abbildung [75]

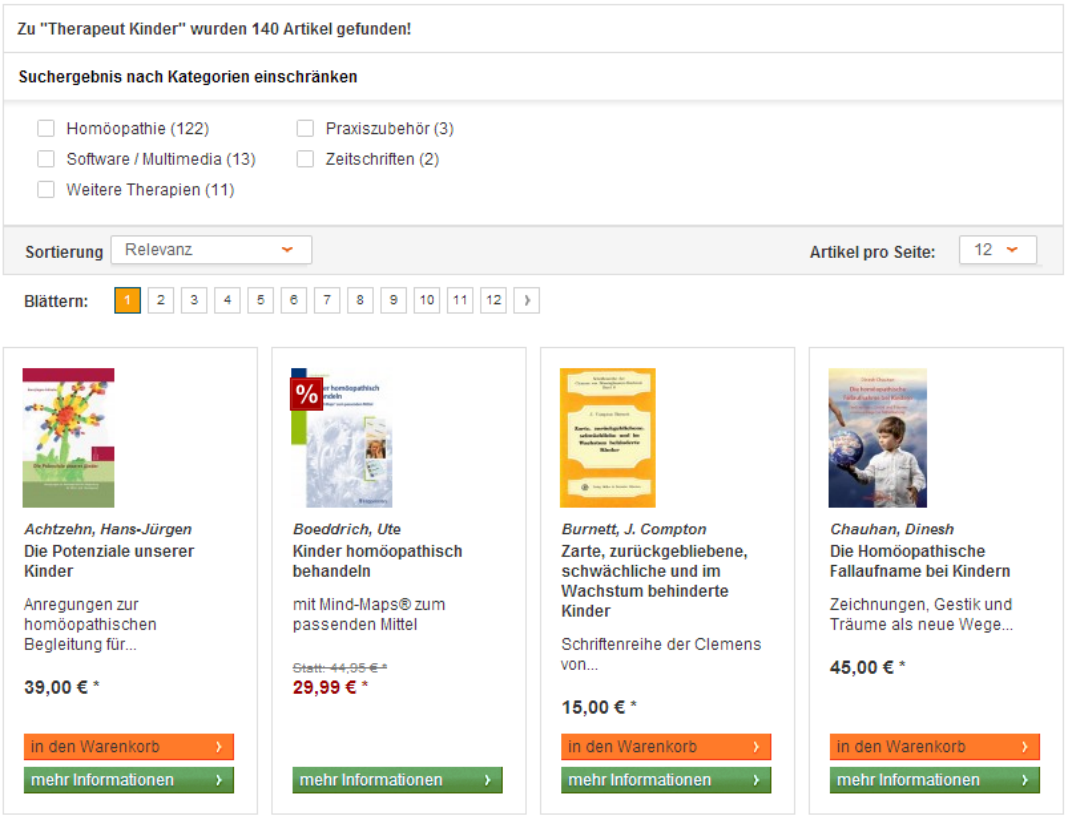


Abbildung [76]

Beim Einstieg über die Suche wird nach Eingabe eine Ajax-basierende Schnellsuche ausgeführt, die - eine sinnvolle Konfiguration der Suche vorausgesetzt, erstaunlich gute Ergebnisse liefert und „onKeyUp“ ausgelöst wird, sofern die festgelegt Mindestanzahl an Zeichen erreicht wird. Ist der gesuchte Artikel nicht bei der Schnellsuche dabei, so kann man mittels „Return“ sowie dem Klicken des Suchicons die vollwertige Suche ausführen.
Diese bietet ebenfalls intelligente Filterfunktionen, die sich automatisch aus der Menge der Suchergebnisse bilden, nur auf Kategorieebene. Dabei kann zunächst auf Hauptkategorieebene gefiltert werden, bei Bedarf auch nach Unterkategorien.

intelligente Suche

Abbildung [77]

Interaktionselemente - Farbleitsystem der Buttons

Abbildung [78]

Um die Bedienung des Shops möglichst intuitiv zu gestalten, wurde folgendes Farbleitsystem umgesetzt: Rottöne für alle Action-Buttons, wobei diejenigen, die eine Transaktion auslösen wie „in Warenkorb legen“ oder „Zahlungspflichtig bestellen“ (a) orange sind, alle weiteren Actionbuttons in rot (b). Buttons, die zu Informationen führen, sind in grün gehalten (c).

12.9. Einrichten der intelligenten Suche

Die intelligente Suche ist ein essentieller Bestandteil eines Shops, der hohe Usability anstrebt. Zielsetzung ist es, dem Shopbetreiber die Möglichkeit zu geben, genau zu konfigurieren, nach welchen Kriterien mit welcher Gewichtung Inhalte gefunden werden.
Diese Suchergebnisse sollen nach Möglichkeit indexiert werden und bei weiteren Suchen berücksichtigt werden, was eine hohe Performance ermöglicht. Shopware bietet zwar immer noch eine „echte“ intelligente Suche als kostenpflichtiges Plugin, aber Shopware CE Version 4.x enthält bereits einen guten Teil der Funktionalitäten, die früher ausnahmslos kostenpflichtig waren.
Konfiguriert wird die Suche so: Zunächst wird die minimale Suchwortlänge festgelegt, für die eine Suche zulässig ist. Die Blacklist für Key-

words, die bei der Suche ignoriert werden sollen (bereits vorkonfiguriert) braucht in der Regel nicht angepasst zu werden.

intelligente Suche

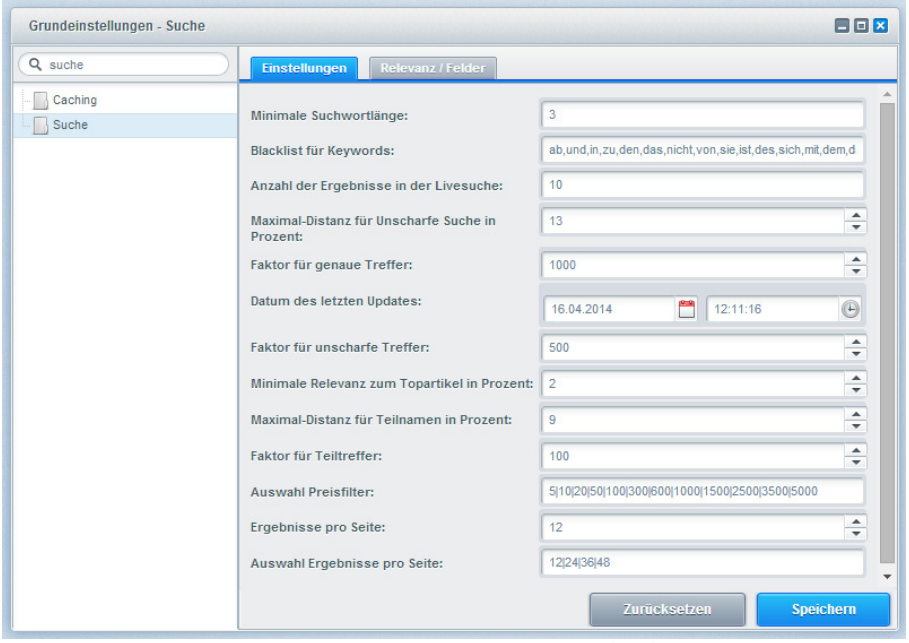


Abbildung „Shopware - intelligente Suche 01“ [58]

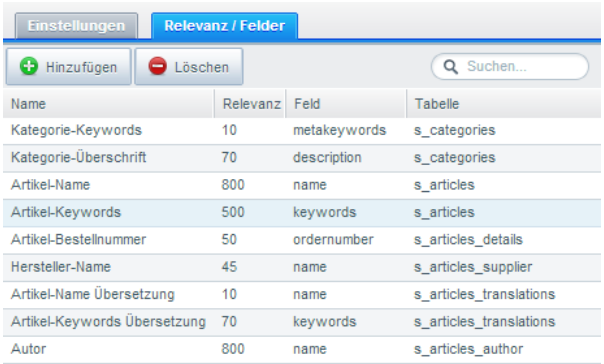


Abbildung „Shopware - intelligente Suche 02“ [58]

Spannend sind Einstellungen, bei denen es um die sog. „Unscharfe Suche“ geht. Diese bietet Algorithmen, mit denen auch Treffer, die nicht unmittelbar mit dem jeweiligen Suchbegriff verknüpft sind, jedoch beispielsweise mit den Top-Treffern eine Verwandschaft aufweisen. Im Reiter „Relevanz/Felder“ sieht man die an einer Suche beteiligten Datenbanktabellen, mittels derer man beliebig viele suchrelevante Felder mit der gewünschten Relevanz versehen kann.

In der Praxis erwies sich diese Architektur als überzeugend einfach, da die sonst aufwändig implementierte Erweiterung des Shops um Buchautoren konnte mit einem Eintrag in der Tabelle „s_search_tables“ in kürzester Zeit umgesetzt werden.
Der Suchindex kann wahlweise per Batch automatisch in regelmässigen Abständen aufgebaut werden, was ebenso manuell angestossen werden kann.

12.10. Datenmigration

Planung

Da eine manuelle „Überführung“ des Datenbestands des alten Shops mindestens aufgrund der hohen Artikelanzahl (2380 Artikel) nicht möglich war, galt es, eine automatisierte Lösung für die Überführung der Daten zu schaffen.

Die Aufgaben der Datenmigration beim Projekt Sunrise waren diese:

- Transfer des existierenden Artikelsortiments in das neue System
- Automatische Überführung der alten Artikelstruktur in Shopware-Kategorien / Shopware-Produkteigenschaften gemäss der mit dem Kunden vereinbarten Migrationsmatrix

Daten-migration

In der Migrationsmatrix wurde - in enger Absprache mit dem Kunden - festgelegt, wie Artikel mit einer bestimmten Kategoriezugehörigkeit im neuen Shop mit welchen Kategorien und Produkteigenschaften geführt wird.

Hauptkategorie alt	Unterkategorie alt	Hauptkategorie neu	Unterkategorie neu	Produktfilter
Displays und Sortiersysteme!		Praxiszubehör	Weiteres Zubehör	Non-Books: Sortiersysteme
Einsteiger und Laien		Bücher	Einführung	Niveau: Einsteiger / Laien
Englische Literatur		Bücher	Englisch	
Hahnemann		Quellen	S. Hahnemann	Theorie: Quellen
Homöopathie-Produkte	Büsten, Miniaturen	Praxiszubehör	Weiteres Zubehör	Non-Books: Büsten / Poster
	Bucheinbände	Praxiszubehör	Weiteres Zubehör	Non-Books: Rund ums Buch
	Ledertaschen	Praxiszubehör	Lederetuis	Non-Books: Lederetuis
	Displays und Sortiersysteme	Praxiszubehör	Weiteres Zubehör	Non-Books: Sortiersysteme
	Mörser	Praxiszubehör	Weiteres Zubehör	Non-Books: Mörser
Kassetten, CDs, Videos	Videos, DVDs	Software / Multimedia	Video-DVDs	Non-Books: Multimedia
	Freiburger Homöopathie Tage	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
	Boller Vorträge	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
	Andreas Krüger	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
	Willibald Gawlik	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
	Sonstige Kassetten, CDs	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
Materia Medica	Enzyklopädien	Bücher	Materia Medica	Niveau: Therapeuten
	Arzneimittellehren	Bücher	Materia Medica	
	Leitsymptome	Bücher	Materia Medica	Theorie: Leitsymptome
	Arzneimittelbeschreibungen	Bücher	Materia Medica	
	Einzelne Arzneimittel	Bücher	Materia Medica	Theorie: Einzelne Arzneimittel
	ältere Werke	Quellen	Weitere Quellen	Theorie: Quellen
Medizin	Heilpraktiker / Ausbildung	Bücher	Medizin	Theorie: Ausbildung (HP) / Weitere Therapien: Heilpraktiker
	Heilpraktiker / Prüfung	Bücher	Medizin	Theorie: Prüfung (HP) / Weitere Therapien: Heilpraktiker
	Weiterführende Literatur	Bücher	Medizin	Weitere Therapien: Medizin / Weitere Therapien: Heilpraktiker
Patientenkarteimappen / Fragebögen				
Naturheilkunde	Phytotherapie	Bücher	Weitere Therapien	Weitere Therapien: Phytotherapie
	Bachblüten	Bücher	Weitere Therapien	Weitere Therapien: Bach-Blüten
	Akupunktur, TCM	Bücher	Weitere Therapien	Weitere Therapien: TCM / Akupunktur
	Biochemie nach Dr. Schüßler	Bücher	Weitere Therapien	Weitere Therapien: Biochemie (Schüßler)
	Ganzheitliche Tiermedizin	Bücher	Weitere Therapien	Praxis: Tiere / Pflanzen
	Allgemeines, Leitfäden	Bücher	Weitere Therapien	Weitere Therapien: Heilpraktiker
	Antlitzdiagnostik	Bücher	Weitere Therapien	Theorie: Anamnese / Weitere Therapien: Heilpraktiker
	Ernährung	Bücher	Weitere Therapien	Weitere Therapien: Ernährung
	Sonstiges	Bücher	Weitere Therapien	

Abb. „Artikelsortiment-Migrationsmatrix (Auszug)“[22]

Dieser Prozess war - auf redaktioneller Ebene - recht zeitauswändig, es vergingen Wochen, bis die finale Vorgehensweise bei der Datenmigration verabschiedet war.

Durchführung

Für die Durchführung der Datenmigration habe ich zunächst eine Mapping-Struktur in Excel erarbeitet, die die IDs - sowohl der Kategorien als auch der Produktfilter - im neuen Shop auflistet:

Kategorie parent	Kategorie parent ID	Kategorie Child	Kategorie Child ID
Bücher	144	Materia Medica	152
		Repertorien	153
		Theorie	154
		Praxis	155
		Einführung	156
		Psychologie	157
		Weitere Therapien	158
		Medizin	159
		Gutscheine	196
		Vorschau	197
		Neuerscheinungen	198
		Sonderpreise	199
		Englische Literatur	200
Quellen	145	C.M. Boger	160
		C.v. Bönninghausen	161
		S. Hahnemann	162
		C. Hering	163
		G.H.G. Jahr	164
		J.T. Kent	165
		M. Mangialavori	166
		R. Sankaran	167
		J. Scholten	168
		G. Vithoulkas	169
		Weitere Quellen	209
Praxiszubehör	147	Arzneibehältnisse	176
		Patientenkarteimappen	178
		Lederetuis	201
		Sunrise-Lederetuis	202
		Sunrise-Stoffrollen	203
		Homö-Set Lederetuis	204
		Weitere Etuis	205
		Weiteres Zubehör	208
Software / Multimedia	148	Radar	179
		Audio-CDs	180
		Video-DVDs	210
Zeitschriften	149	AHZ	182
		ZKH	183
		Homöopathie Konkret	184
		Homöopathie Zeitschrift	185
		Spektrum der Homöopathie	186
		Homoeopathia Viva	187
		Tierhomöopathie	188
		Ganzheitliche Tierhomöopathie	189
		DHZ	190

Abbildung „Migrations-Mapping (Kategorien)“ [23]

Daten-migration

Filter	Filter ID	Filter-Option	Filter-Option ID	Filter-Attribut	Filter-Attribut ID
Literatur	12	Praxis	33	Fallsammlungen	172
				Frauen	168
				Kinder	167
				Medizin	173
				Onkologie	170
				Psychologie	169
				Ratgeber	166
				Therapie (Allgemein)	164
				Therapie (Kopf -> Fuß	165
				Tiere / Pflanzen	171
		Sunrise Bücherregal	36	Einsteiger / Laien	178
				Studium / Ausbildung	179
				Therapeuten	180
		Weitere Therapien	37	Bach-Blüten	187
				Biochemie (Schüßler)	185
				Ernährung	183
				Geist und Seele	186
				Heilpraktiker	181
				Impfung	189
				Medizin	182
				Phytotherapie	184
				TCM / Akupunktur	188
		Theorie	38	Anamnese	192
				Arzneiprüfungen	196
				Ausbildung (HP)	203
				Biografien	201
				Einzelne Arzneimittel	197
				Geschichte / Forschung	199
				Handbücher (AML)	194
				Lehrbücher	190
				Leitsymptome	193
				Lernhilfen	198
				Miasmen	191
				Prüfung (HP)	204
				Quellen	200
				Romane	202
				Vorlesungen (AML)	195
		Sonderpreise	35	Vorschau	174
				Neuerscheinungen	175
				Preisaktionen	176
				Antiquariat	177
Non-Books	13	Non-Books	40	Alu-Etuis	208
				Arzneibehältnisse	206
				Arzneimittelbriefchen	207
				Büsten / Poster	212
				Lederetuis	209
				Mörser	213
				Multimedia	215
				Patientenkarteimapper	210
				Rund ums Buch	214
				Software	216
				Sortiersysteme	211
				Sunrise Produkte	205

Daten-migration

Abbildung „Migrations-Mapping (Produkteigenschaften / Produktfilter)“ [24]

Auf der Grundlage dieser Mappings sowie der übergeordneten Migrationsmatrix für das Artikelsortiments erstellte ich logisch zugehörige Datenbanktabellen. Diese dienten dazu, mittels SQL die Daten des alten Shop so abzufragen, dass die für den Import in den neuen Shop benötigen CSV-Dateien aus den Abfrageresultaten erzeugt werden konnnten.

Die Struktur der Datenbanktabellen für die Migration ist diese:

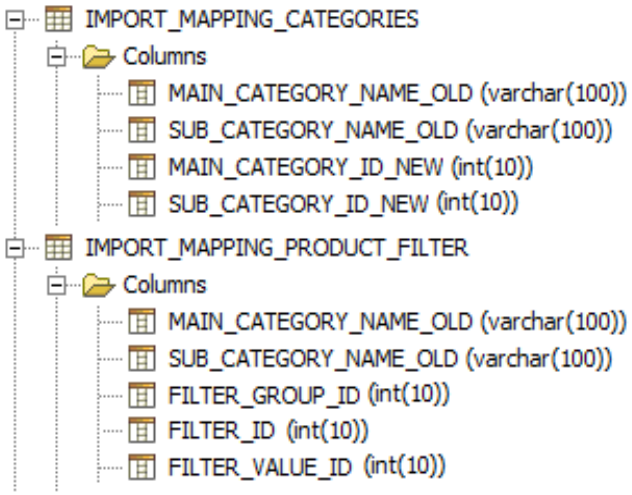


Abbildung „Datenbanktabellen für Migration“ [26]

Daten-migration

Ursprünglich hatte ich geplant, die beiden Tabellen direkt in die Datenbankstruktur des alten Shop einzufügen.
Da dieser jedoch auf mittels einer ORACLE 8i Datenbank realisiert war, die nur einen JOIN pro Tabelle zulässt, habe ich sowohl den Datenbestand des alten Shops wie auch die Tabellen für die Migration in die Datenbank des neuen Shops, mySQL, überführt.

```
SELECT  CONCAT('SUN',CAST(p.id as CHAR)) as ordernumber,
        CASE
            WHEN subcat1.label='Sunrise-Lederetuis' then 'Sunrise'
            WHEN subcat1.label='Sunrise-Stoffrollen' then 'Sunrise'
            ELSE 'Sunrise Literatur'
         END
         as supplier,

        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.title,'?',''),CHAR(10),''),CHAR(13),''),';','')) as name,
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.subtitle,'?',''),CHAR(10),''),CHAR(13),''),';','')) as description,
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.web_comments,'?',''),CHAR(10),''),CHAR(13),''),';','')) as description_long,
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.keywords,'?',''),CHAR(10),''),CHAR(13),''),';','')) as keywords,
        TRIM(REPLACE(REPLACE(REPLACE(a.name,'?',''),'MÄNGELEXEMPLAR',''),';',''))  as author,//Beseitigen von Datenqualitätsfehlern
        1 as active,
        19 as tax,
        CAST(p.price AS DECIMAL(10,2)) as price,
        CAST((p.price /( 19 / 100 + 1))AS DECIMAL(10,2)) as net_price,

        9 as unitID,

        10000 as instock,
        0 as stockmin,

        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.EDITION,'?',''),CHAR(10),''),CHAR(13),''),';','')) as attr_attr2,//Auflage
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.BOOK_DATA,'?',''),CHAR(10),''),CHAR(13),''),';','')) as attr_attr3, //Seitenzahl
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.TEASER,'?',''),CHAR(10),''),CHAR(13),''),';','')) as attr_attr4,//Untertitel
        TRIM(REPLACE(REPLACE(REPLACE(REPLACE(p.subtitle,'?',''),CHAR(10),''),CHAR(13),''),';',''))  as attr_attr5,
        CONCAT(map_cat1.MAIN_CATEGORY_ID_NEW,CONCAT('|',map_cat1.SUB_CATEGORY_ID_NEW)) as categories,
        CAST(map_filter1.FILTER_ID as unsigned) as filtergroupID,
        CAST(map_filter1.FILTER_VALUE_ID as unsigned)  as propertyValue
FROM
        PRODUCTS p
        LEFT JOIN AUTHORS a on p.author_id = a.id
        LEFT JOIN CATEGORIES cat1 on  SUBSTRING(p.CATEGORY1_ID,1,2)= cat1.MAIN_CATEGORY
                                      AND cat1.SUBCATEGORY1 = 0  AND cat1.SUBCATEGORY2 = 0
        LEFT JOIN CATEGORIES subcat1 on SUBSTRING(p.CATEGORY1_ID,1,2)= subcat1.MAIN_CATEGORY
                                      AND SUBSTRING(p.CATEGORY1_ID,3,1)= subcat1.SUBCATEGORY1 AND 0 = subcat1.SUBCATEGORY2
        LEFT JOIN IMPORT_MAPPING_CATEGORIES  map_cat1 on  trim(map_cat1.MAIN_CATEGORY_NAME_OLD) = trim(cat1.label)
                                         AND trim(map_cat1.SUB_CATEGORY_NAME_OLD) = trim(subcat1.label)
        LEFT JOIN IMPORT_MAPPING_PRODUCT_FILTER  map_filter1 on  trim(map_filter1.MAIN_CATEGORY_NAME_OLD) = trim(cat1.label)
                  AND trim(map_filter1.SUB_CATEGORY_NAME_OLD) = trim(subcat1.label)

WHERE CATEGORY1_ID <= 0 AND CATEGORY2_ID >  0
ORDER BY cat1.label
```

Abbildung „SQL Datenmigration“ [25]

Dadurch konnten alle Operationen innerhalb einer Datenbank durchgeführt werden, was das Prozedere vereinfachte.

Ziel dieser Vorgehensweise war die automatisierte Erstellung von CSV- Dateien für den Import ins neue System, sowohl (wie in Abbildung „SQL Datenmigration“) für Artikeldaten wie auch für Artikelbilder, die ebenfalls mittels SQL-basierten CSV Dateien mit Daten bestückt wurden:

```
ordernumber;image;main;description;position
SUN11097;http://www.sunrise-versand.de/prodpix/gTN_Ploog_Kinder_Traumata.jpg;1;;1
SUN12487;http://www.sunrise-versand.de/prodpix/gTN_Scheffer_B-B-Therapie%20Irisi.jpg;1;;1
SUN12491;http://www.sunrise-versand.de/prodpix/gTN_Scheffer-Storl_Seelenpflanz.jpg;1;;1
SUN14319;http://www.sunrise-versand.de/prodpix/gTN_Dimitriadis_Chron.Krankheiten.jpg;1;;1
SUN14578;http://www.sunrise-versand.de/prodpix/gTN_Liem_Morphodynamik.jpg;1;;1
SUN15322;http://www.sunrise-versand.de/prodpix/gTN_Sonnenschmidt_Gehirn.jpg;1;;1
SUN15408;http://www.sunrise-versand.de/prodpix/gTN_Wacker_Heilpflanzen.jpg;1;;1
SUN18575;http://www.sunrise-versand.de/prodpix/gTN_Tyler_Krankheitszustaende_2.A.jpg;1;;1
SUN18576;http://www.sunrise-versand.de/prodpix/gTN_Hainbuch_Bienen.jpg;1;;1
SUN18577;http://www.sunrise-versand.de/prodpix/gTN_Koeslin_Pruefungstraining.jpg;1;;1
SUN18579;http://www.sunrise-versand.de/prodpix/gTN_Sehgal_Platina_CD.jpg;1;;1
SUN18580;http://www.sunrise-versand.de/prodpix/gTN_Sehgal_Cocculus_CD.jpg;1;;1
SUN18581;http://www.sunrise-versand.de/prodpix/gTN_Sehgal_Belladonna_CD.jpg;1;;1
```

(Auszug CSV Datei „Produktbilder“ - CSV Dateien für den Import von Artikeln / Kategorien+Produktfilter sind im Anhang beigefügt.)

13. Entwicklung eines Backend-Plugins für E-Book-Import

Gleich vorweg: ohne Hilfestellung seitens des Herstellers in Form von Tutorials hätte ich diese Aufgabe nicht bewältigen können:

Shopware 4 - Grundlagen der Plugin Entwicklung
http://wiki.shopware.de/_detail_971_867.html

Entwicklung von Backend-Modulen
http://wiki.shopware.de/Entwicklung-von-Backend-Modulen_detail_818.html

Backend Plugin

Plugin-Entwicklung: Boilerplate-Code (kommentiert und unkommentiert)
http://wiki.shopware.de/Plugin-Entwicklung-Boilerplate-Code_detail_1077_869.html

Shopware Backend Komponenten - Batch Prozesse
http://wiki.shopware.de/Shopware-Backend-Komponenten-Batch-Prozesse_detail_1421.html

Ich hatte zu Beginn keine Erfahrung mit einem der beteiligten Frameworks, erst recht nicht mit den Wechselwirkungen der Elemente untereinander sowie dessen shopwareseitige Erweiterungen. Der Einstieg bei der Programmierung fiel mir, trotz allem angelesenen Vorwissen, recht schwer.
Zu leicht brachte man sich mit minimal fehlerhaftem Code in eine Situation, bei der die Ausführung des Codes an einer bestimmten Stelle abbrach, aber danach man nicht mehr sagen konnte, ob weiteres Fehlverhalten vom Code ausgelöst wurde, oder von den Proxies - stetiges Cache-Leeren per Kommandozeile in Kombination mit Aus-/Einloggen war notwendig.
Das Gleiche galt für den Umgang mit dem Debug-Werkzeug „XDebug“ in Verbindung mit meiner IDE. Zwar lieferte dieses umfangreiche und gut strukturierte Daten zum Debuggen, jedoch brach die Verbindung auf Port 3000 zur PHP Storm immer wieder ab, so dass die IDE neu gestartet werden musste.
Was am schwersten fiel, war dies: Einzuschätzen, was jeweils zwischen den einzelnen Frameworks stattfindet, was mindestens bei rein logischen Fehlern wichtig ist. Die meisten dieser Fehler wurden seitens Doctrine geworfen und betrafen Aspekte der Persistenz, also des festen Schreibens von Daten im Model-System. Die Syntax von Ext JS ist - sobald es über einfache Beispiele hinausgeht, wenig intuitiv und gewöhnungsbedürftig.
Meine anfängliche Motivation, mir im Rahmen der Pluginprogrammierung die Grundlagen des Versionsmanagementsystem anzueignen, ist nach ein paar Wochen geschwunden: Mehrere Male konnte ich nur über Ordnervergleiche des Plugins mit der vollständigen Vorgängerversion erschliessen, welches die jeweils kritische Codestelle war, wenn etwas nicht (mehr) funktionierte. Doch nun zum Plugin:

13.1. Zieldefinition

Bei der Entwicklung des Plugins sollte zum einen ein Prototyp für ein praxistaugliches Werkzeug für den Import von E-Book-Daten aus dem Artikelsortiment geschaffen werden, zum anderen sollte ein „handfester“ Eindruck beim Umgang mit den vorgestellten Technologien gewonnen werden, wie auch ein Know-How Zuwachs beim Programmieren mit den beteiligten Frameworks.
Ziel des konkreten Plugins ist es, den E-Book-Artikelbestand des Buchgroßhändlers KNV (http://www.knv.de) für die Verwendung in einem Shopware-E-Shop zugänglich zu machen. Dabei soll der Shopbetreiber auswählen können, welche Titel er in sein Artikelsortiment aufnehmen bzw. aktualisieren möchte. Die Entwicklung des Plugins hat exemplarischen Charakter - so lag die Zielsetzung nicht darin, ein marktfähiges Plugin zu erstellen, sondern in der konkreten Anwendung des Gelernten sowie einer näheren Betrachtung der Shopware-Elemente.
Das Plugin übernimmt dabei den reinen Importvorgang - nicht jedoch die Transaktionsverwaltung beim Anfordern des E-Book-Downloads bei KNV. Der Anbieter KNV bietet zwar ein Testinterface für diesen Zweck, mit dem jedoch nur ein einziger Artikel abrufbar ist - damit kann das formale Funktionieren des Download-Anforderns getestet werden, mehr nicht. Da seitens Sunrise kein konkreter Nutzungsvertrag der KNV Dienstleistungen vorliegt, musste ich aus ähnlichen Gründen den Import der Produktbilder simulieren - dazu später mehr.

Backend Plugin

13.2. Funktionsbeschreibung

Das Plugin „EQX Ebooks (KNV)“ wird im Backend über den Plugin Manager installiert - zu finden ist es unter „Lokale Erweiterungen“.
Beim Installationsvorgang wird der Menüpunkt „Equinoxe Ebooks - KNV“ ins Backend eingehängt sowie das exemplarische XML File mit den KNV Ebook-Daten eingelesen. Exemplarisch deshalb, weil ich die mehreren Tausend Titel, die seitens KNV bereitstehen, auf eine sinnvolle Schnittmenge heruntergebrochen - zum einen, um Tests zu erleichtern, zum anderen aus redaktionellen Gründen. Dies wurde ausserhalb des Plugins mittels PHP bewerkstelligt - ich habe dabei die Aufgabe des Shopbetreibers übernommen und festgelegt, dass Artikel, die eines der folgenden Keywords enthalten, für einen Import in Frage kommen: „Medizin, Ratgeber, Krank, Anatomie, Arznei, Askese, Buddhismus, Psychologie, Ethik, Genmanipulation, Gesundheit, Kreativität, Myofunktionell, Nachhaltigkeit, Neurologie, Onkologie, Pflanzen, Pflege,Psychiatrie, Psycholinguistik,Psychotherapie,Schlaf,

Abbildung [79]

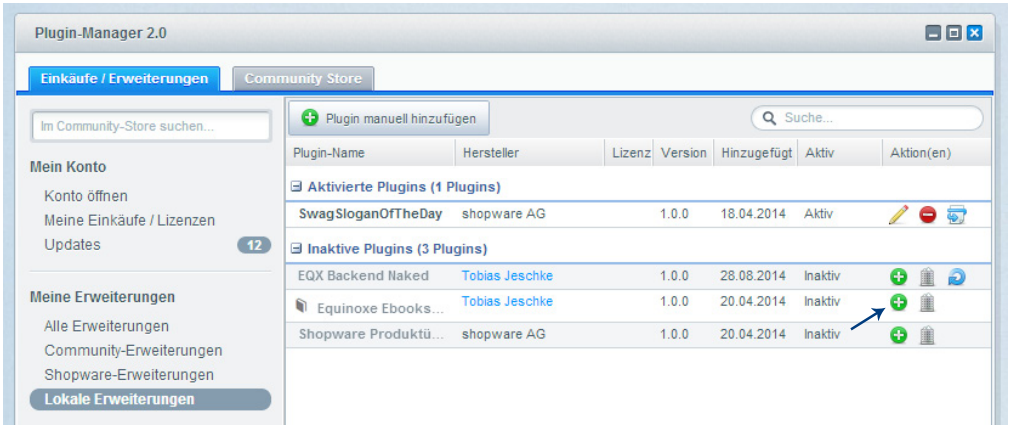


Abbildung [79]

Backend Plugin

Schmerztherapie,Sexualität“. Dabei habe ich bewusst auch Titel aufgenommen, die nicht unmittelbar mit Homöopathie zu tun haben (z.B. Ratgeber für „Stricken“ oder Bücher über Energieeffizienz) - der Gedanke dabei war, das Angebot von www.sunrise-versand.de „sanft“ zu erweitern. In der Anlage habe ich im Ordner „KNV“ sowohl die XML-Spezifikation für den ONYX-basierenden Artikelkatalog wie auch die beim Import verwendeten XML Dateien beigefügt - „20140319_03688646_OUSS.xml“ enthält den vollständigen Katalog, „my_data_handbag.xml“ die im Plugin verwendete Katalogteilmenge. Wurde das Plugin erfolgreich installiert (und aktiviert), so steht - nach erneutem Login - unterhalb von „Artikel“ der Menüpunkt „Equinoxe EBooks - Anbindung an KNV“ zur Verfügung.

Abbildung [80]

Dieser listet die aus der XML-Datei ausgelesenen Daten mittels ausgewählter, sortierbarer Spalten auf, mittels Blätterfunktion oder dem Erhöhen des „Einträge pro Seite“-Wertes können diese in Augenschein genommen werden. Die eigentliche Importfunktionalität ist einfach: der Shopbetreiber entscheidet, ob ein Titel ins Artikelsortiment

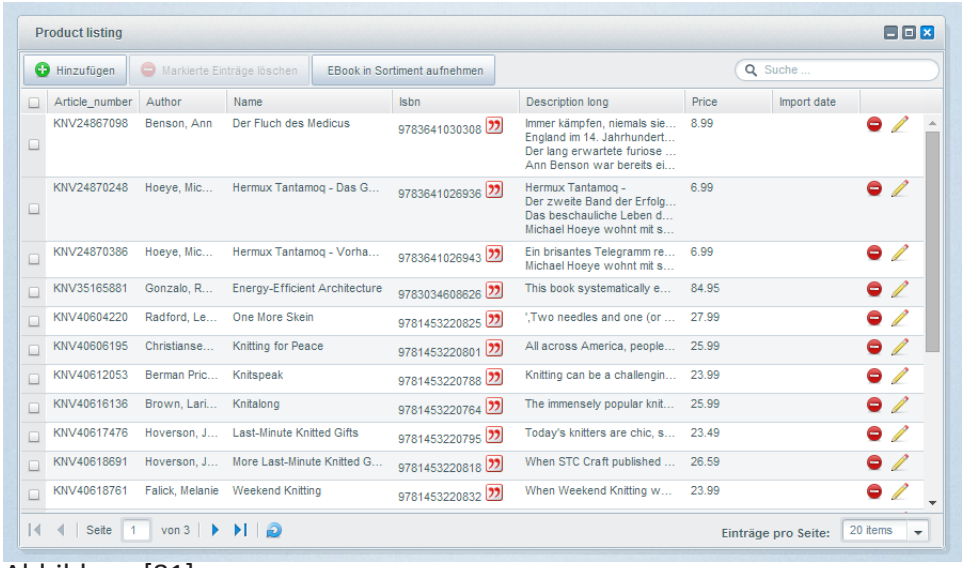
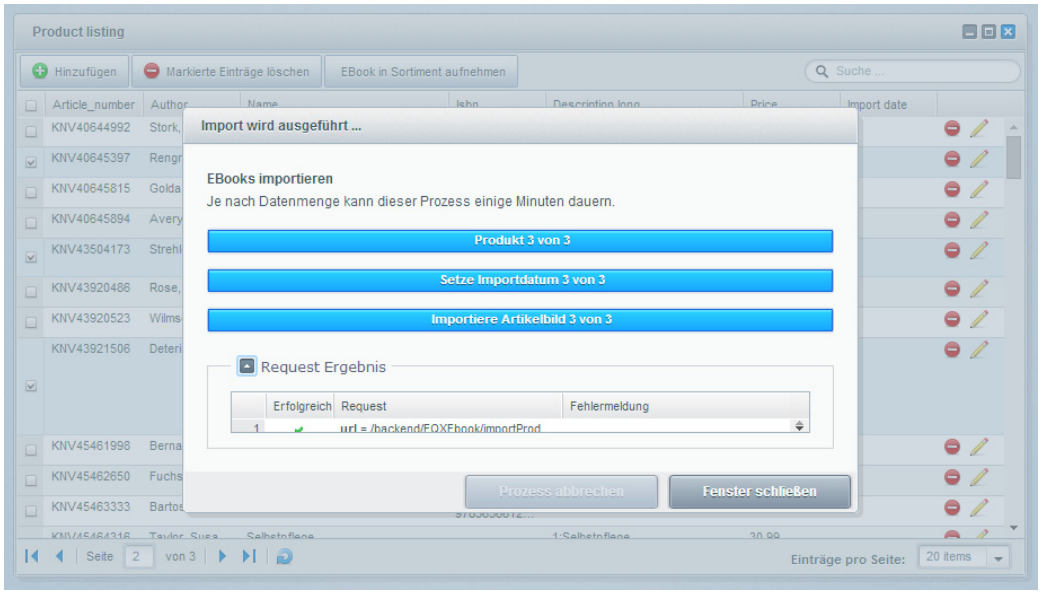


Abbildung [81]

aufgenommen werden soll, hakt die Checkkboxen für die zu importierenden Titel an und klickt auf „EBook ins Sortiment aufnehmen“.



Backend Plugin

Abbildung [82]

Das Plugin führt dann den eigentlichen Importvorgang für die gewählten Titel aus - übernimmt diese also in das Artikelsortiment des Shops - und versieht die zugehörigen KNV Datensätze mit dem Importdatum: Auf diese Weise kann der Shopbetreiber (Fenster hierfür neu laden) sehen, welche Datensätze er bereits importiert hat und wann der letzte Import / das letzte Update für einen Artikel ausgeführt wurde.

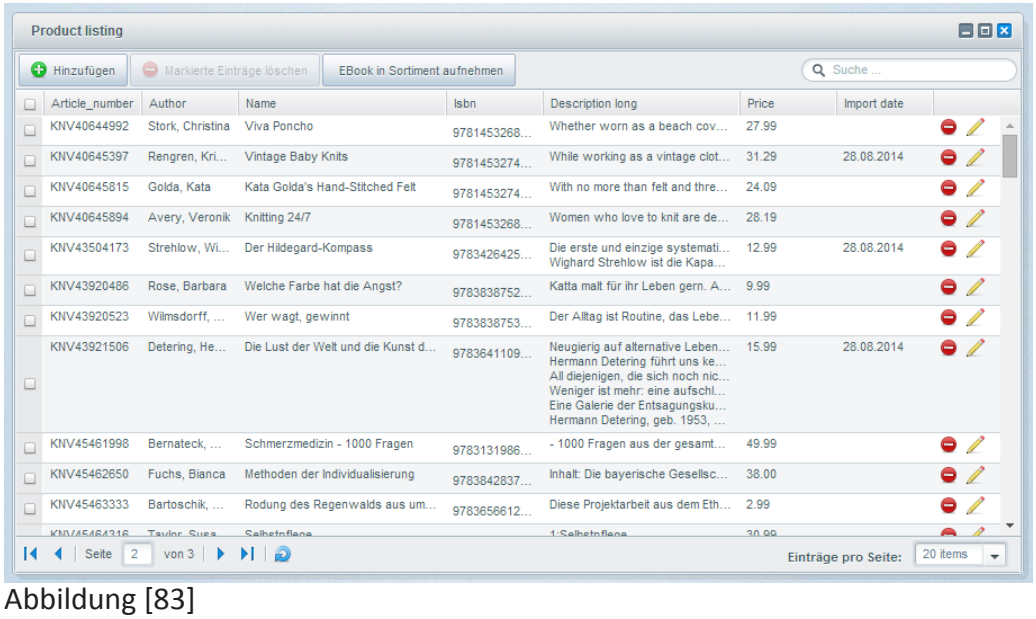


Abbildung [83]

Wichtig: Damit das Plugin auch ausserhalb des - mittels Autorenverwaltung im Kernsystem modifizierte Shops bei www.sunrise-versand.de getestet und zu Anschauungszwecken dienen kann, habe ich den Autor ausser acht gelassen - bei Bedarf ist dies schnell implementiert, genau wie Seitenzahlen oder andere Elemente, die je nach Bedarf des Shopbetreibers benötigt werden.

13.3. Implementierung

Bootstrap

Damit nachvollziehbar wird, wie das Plugin intern funktioniert, möchte ich anhand eines Code-Reviews Stück für Stück erläutern, aus welchen Teilen es besteht und wofür diese zuständig sind. Dabei beziehe ich mich auf die numerisch ausgezeichneten Dateien der Grafik „Plugin - Dateistruktur“. Zusätzlich habe ich innerhalb der eigentlichen Dateien mit vielen Kommentaren anschaulich gemacht, was jeweils geschieht.

Backend Plugin

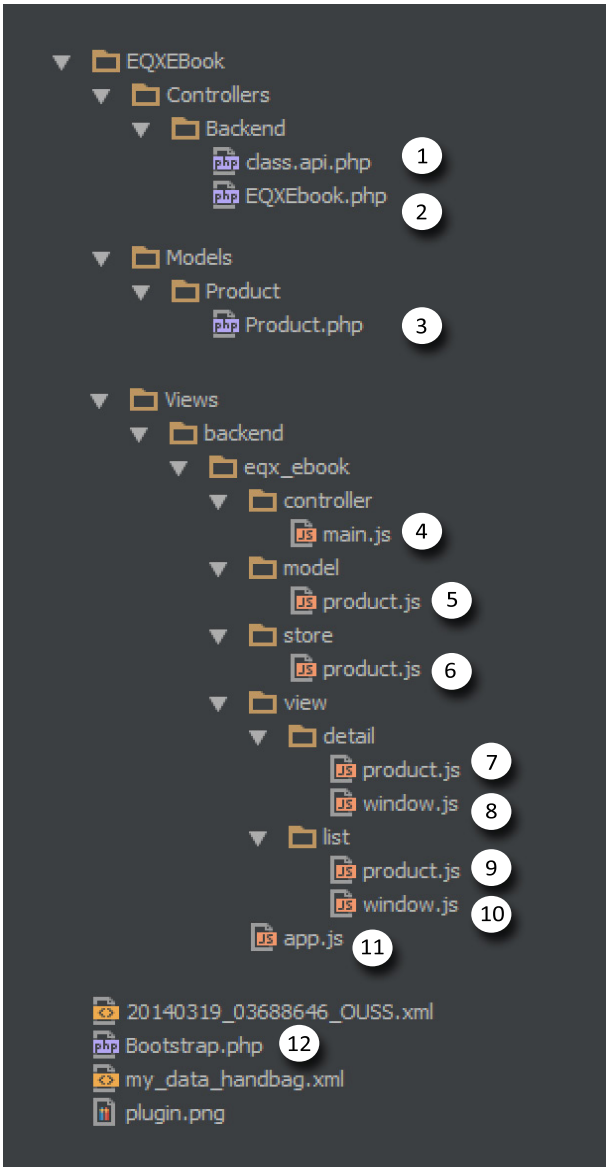


Abbildung „Plugin - Dateistruktur“ [74]

Damit ein Plugin seitens des Plugin-Managers verarbeitet werden kann, gilt es, gewisse Formalitäten einzuhalten - dies betrifft vornehmlich die allen Plugins gemeinsame Datei Bootstrap.php (12). Diese muss mindestens folgende Funktionen enthalten:

- *getCapabilities() //was darf das Plugin [install/update/enable] ?*
- *getLabel() // wie heisst es ?*
- *getVersion() //welche Version ?*
- *getInfo() // Rückgabe der o.g. Infos, zusammengefasst*
- *install() //wird bei der Installation aufgerufen*

In der install()-Methode habe ich einen Event registriert, der den Plugin-Controller im System anmeldet:

```
$this->subscribeEvent(
        'Enlight_Controller_Dispatcher_ControllerPath_Backend_EQXEbook',
        'getBackendController'
);
```

Dies bedeutet, dass /backend/EQXEBook nun als Controller-Pfad zur Verfügung steht, die jeweiligen Actions werden per Unterordner aufgerufen, also /backend/EQXEBook/[NAME_DER_ACTION].
Ich werde dies später am konkreten Beispiel zeigen.

Der Menüeintrag wird generiert und das sog. „Custom Model“ wird in Doctrine registriert:

```
$this->updateSchema();

protected function updateSchema()
 {
     //perform a custom model registration
     $this->registerCustomModels();//a.

     $em = $this->Application()->Models();
     /*Doctrine‘s schema tool provides functionality to create both Model
     AND corresponding Database table*/
     $tool = new \Doctrine\ORM\Tools\SchemaTool($em);//b.

     //read defined Model attributes and methods
     $classes = array(
        $em->getClassMetadata(‚Shopware\CustomModels\Product\Product‘)//c.
     );

     try {
        $tool->dropSchema($classes);
     } catch (Exception $e) {
        //ignore
     }
     //(re-)create Scheme
     $tool->createSchema($classes);//d.
 }
```


Zunächst wird Doctrine mitgeteilt, dass ein Custom Model, also ein selbstgestaltetes Model registriert werden soll (a.). Dann wird eine Instanz von Doctrine‘s Schematool erzeugt (b.) - dieses ist für das automatische Anlegen von Datenbanktabellen zuständig (wie in Kapitel 9.1.4. beschrieben). Anhand der hinterlegten Metadaten des Models (c) und wird das Schema erstellt (d), ggf. wird es vorher gelöscht, um es danach neu erzeugen zu können.

Das Custom Model

In der Datei Product.php (3) finden sich die gerade angesprochenen Metadaten, mittels derer Tabellenname, die Klassenattribute sowie zugehörige Datenbankspalten definiert werden. Dies geschieht in einem eigenen Namespace: *namespace Shopware\CustomModels\Product; .*
Der Namespace Shopware\CustomModels ist dabei ausschliesslich für selbsterzeugte Models innerhalb Plugins reserviert.

Auf Controllerebene ist festzuhalten, dass alle Actions des Plugins in einer einzigen PHP-Datei gebündelt werden - diese muss exakt so benannt sein wie der Ordnername des Plugin (2).
Am Anfang der Datei fällt auf, dass das Shopware-Model „Article“ eingebunden ist:

```
use Shopware\Models\Article\Article as ArticleModel;
```

Dies wird dafür benötigt - dass auf Funktionen des Standard-Article-Models zugegriffen werden kann.
Die Controller-Methoden sind jeweils „public“, damit diese von außen aufgerufen werden können. Dabei stehen folgende Methoden zur Verfügung:

Backend Plugin

1. *importProductsAction()*
   Diese Methode ist die „eigentliche“ Importmethode, die bei Aufruf als erstes eine Instanz der internen Shopware-API erstellt:
   $articleResource =
   \Shopware\Components\Api\Manager::getResource(‚Article‘);

   Als nächstes werden die ProductIDs des ausgewählten EBooks ausgelesen und die Daten dieses Produkts angezogen.
   Dies bezieht sich auf die Gesamtheit der importierfähigen Produkte,
   also der Auswahlliste, was man importieren möchte.

```
$productId = $this->Request()->getParam(‚productId‘);
$product = $this->getManager()->find(
        $this->model,
        $productId
);
```

   Als nächstes wird ermittelt, ob im Shop die Kategorie „E-Books“ angelegt ist - ist dies der Fall, so werden die Artikel beim Import automatisch in diese abgelegt - falls nicht, landen diese in der Hauptkategorie „Deutsch“.

```
$categoryName = „E-Books“;
$builder = Shopware()->Models()->createQueryBuilder();
        $builder->select(array(‚category‘))
           ->from(‚Shopware\Models\Category\Category‘, ‚category‘)
           ->where(‚category.name = :categoryName‘);
         $builder->setParameter(‚categoryName‘, $categoryName);
        $categoryEBookId = $builder->getQuery()->getOneOrNullResult();
```

   Auf der nächsten Seite ist zu sehen, wie der Array für das Erstellen eines Artikels gebildet wird - dabei habe ich mich an den Herstellerbeispielen für die REST-API orientiert - die Struktur war übertragbar. Zusätzlich habe ich Schritt für Schritt mittels XDebug geprüft, an welcher Stelle der Import in einen Fehler läuft und so Rückschlüsse auf die Beschaffenheit des erwarteten Arrays gezogen.

```
//set Article data array
$myArticle = array(
  ‚name‘ => $product->getName(),
  ‚active‘ => true,
  ‚tax‘ => 19,
  ‚supplier‘ => ‚KNV‘,
  ‚description‘ => $product->getDescription(),
  ‚descriptionLong‘ => $product->getDescriptionLong(),
  ‚categories‘ => array(
          array(‚id‘ => $categoryEBookId),
  ),
  ‚images‘ => array(
      array(‚link‘ => ‚http://lorempixel.com/g/400/200/‘),
      array(‚link‘ => ‚http://lorempixel.com/g/400/200/‘),
  ),
  ‚mainDetail‘ => array(
      ‚number‘ => $product->getArticleNumber(),
      ‚instock‘=> 1000,
      ‚stockmin‘=> 0,
      ‚prices‘ => array(
          array(
              ‚customerGroupKey‘ => ‚EK‘,
              ‚price‘ => $product->getPrice(),
          ),
      )
  )
);
```

Backend Plugin

Im „images“-Array sieht man, wie ich Ersatz für den nur als Vertragskunde verfügbaren Artikelbild-FTP-Zugang geschaffen habe.
Der simple, aber wirkungsvolle Dienst von lorempixel.com, mittels Aufruf der URL „http://lorempixelcom/g/400/200/“ ein Zufallsbild der Grösse „400 x 200 px“ abzuholen, leistet gute Dienste.

Abschliessend wird der Artikel erstellt:

```
$article = $articleResource->create($myArticle)
```

Die weiteren Action-Methoden des Controllers, changeCreateDateAction() sowie importPictureAction() dienten hauptsächlich dazu, ein Gefühl für die Controller-Architektur von Shopware zu bekommen, wie auch für einen Plugin-Benutzer transparent zu machen, was beim Import konkret geschieht. Dabei gilt es zu beachten, dass die Zuweisungen nur dann persistent in die Datenbank geschrieben werden, wenn dem Entity-Manager dies mittels flush() mitgeteilt wird.Im Prinzip setzt die importProductsAction() bereits alles um, was benötigt wird:

```
$product->setImported(1);
$product->setImportDate(now);
```

Die Methode „importPictureAction()“ ist dafür vorbereitet, weitere Aktionen unter Verwendung der REST-API durchzuführen und dient der potentiellen Erweiterung des Plugins. Sie könnte ebenso „doSomethingUseFul()“ heissen.

Die View - Ext JS

Ext JS organisiert seine View-Dateien intern wiederum mit dem MVC-Entwufsmuster, wobei Ext JS zusätzlich mit dem sog. „Store“ arbeitet, dem im Plugin jedoch eine rein formale Rolle zukommt, weshalb ich nicht weiter darauf eingehen möchte.
Der „Körper“ einer Ext JS Komponente ist die app.js (11).

Backend Plugin

```
//define subapplications extending default backend functionality
Ext.define(‚Shopware.apps.EQXEbook‘, {
  extend: ‚Enlight.app.SubApplication‘,

  //Class Name should refer to the PHP Controller Name
  name:‘Shopware.apps.EQXEbook‘,
  loadPath: ‚{url action=load}‘,
  bulkLoad: true,
  controllers: [ ‚Main‘ ],
  views: [
    ‚list.Window‘,
    ‚list.Product‘,
    ‚detail.Product‘,
    ‚detail.Window‘
  ],
  models: [
    ‚Product‘
  ],
  stores: [ ‚Product‘ ],
  //launch method triggers main controller
  launch: function() {
    return this.getController(‚Main‘).mainWindow;
  }
})
```

Diese erstellt eine EQXEBook-Applikation, abgeleitet von den „Enlight-SubApplications“. Der Klassenname muss dabei dem Namen des PHP-Controllers entsprechen.
Elemente dieser Ext JS - Instanz sind:

- die verschiedenen Views, wobei die Fenster (also Ext JS Grids) unabhängig vom jeweiligen Inhalt angegeben werden müssen. Ansichten gibt es für die Produktliste des Importauswahlfensters sowie für die Einzelansicht eines EBooks.

Innerhalb der View wird auch der Button für die Aktion „E-Book in Sortiment aufnehmen“ definitiert sowie dessen Event registriert (9), was onClick vom Controller (2) ausgewertet wird .

```
createToolbarButton: function() {
 var me = this;
 return Ext.create(‚Ext.button.Button‘, {
    text: ‚EBook in Sortiment aufnehmen‘,
            handler: function() {
       me.fireEvent(‚change-products‘, me);//register Event for custom button
    }
 });
}
```

Backend Plugin

- Ein Model „Product“ - zuständig dafür, welche Teile des Doctrine Custom Models von Ext JS verarbeitet werden sollen
- Der Main-Controller (das „Gehirn“)

Diesem Teil möchte ich abschliessend widmen - der Datei main.js (4).

```
// Naming convention: Class Name should refer to the PHP Controller name
Ext.define(‚Shopware.apps.EQXEbook.controller.Main‘, {
  extend: ‚Enlight.app.Controller‘,/*initialize listing window (property „mainWin
          dow“ must be set - this implies f.e. the standarized „window-close-But
          ton“)/*
  init: function() {
    var me = this;    --> dies bindet „this“ an die variable me, so dass „this“  auch in Unter-
                          funktionen verwendet werden kann
    me.control({
      ‚product-listing-grid‘: { -> das „change-products-Event  wird gefangen und die Funktion
                                  „displayProcessWindow“ aufgerufen
        ‚change-products‘: me.displayProcessWindow/*get Event „change-products“, execute
                                                    function „displayProcessWindow“*/
      },
      ‚shopware-progress-window‘: {
        ‚import-products-process‘: me.onImportProducts,
        ‚change-create-date-process‘: me.onChangeCreateDate,
        ‚import-picture-process‘: me.onImportPicture
      }
    });
    me.mainWindow = me.getView(‚list.Window‘).create({ }).show();
  },

  /*register event listener for product deactivation
  task = current executed task
  record = current product record
  callback = callback for current iteration - if no callback operation is executed,
                                              only the first pro ductwill be treated*/
```

Backend Plugin

```
onImportProducts: function (task, record, callback) {
    Ext.Ajax.request({ -> Ext JS erledigt die Kommunikation mit dem Webserver via AJAX Requests

        url: ,{url controller=EQXEbook action=importProducts}‘, -> dieser ruft die zugehörige
                                                                    Controller Action auf
        method: ‚POST‘,
        params: {
            productId: record.get(‚id‘)
                    --> und übergibt die jew. Product-ID als Parameter
        },
        success: function(response, operation) {
            callback(response, operation);
                -> damit nicht nur ein Produkt abgearbeitet wird, wird nach erfolgreicher Ausführung
                    der Controller-Action die übergebene Callback-Funktion ausgeführt, bis keine
                    Produkte mehr „übrig“ sind
        }
    });
},

-> weitere Methoden können nach dem gleichen Prinzip eingehängt werden:
onChangeCreateDate: function (task, record, callback) {
    Ext.Ajax.request({
        url: ,{url controller=EQXEbook action=changeCreateDate}‘,
        method: ‚POST‘,
        params: {
            productId: record.get(‚id‘)
        },
        success: function(response, operation) {
            callback(response, operation);
        }
    });
},
onImportPicture: function (task, record, callback) {
    Ext.Ajax.request({
        url: ,{url controller=EQXEbook action=importPicture}‘,
        method: ‚POST‘,
        params: {
            productId: record.get(‚id‘)
        },
        success: function(response, operation) {
            callback(response, operation);
        }
    });
},

displayProcessWindow: function(grid) {
     --> die im Grid selektierten Datensätze werden ermittelt
    var selection = grid.getSelectionModel().getSelection();//retrieve selected product records

   if (selection.length <= 0) return;//exit function when no products are selected
      //if products are selected, trigger progress window via ExtJs
```

-> und sofern diese vorhanden sind, generiert Ext JS das Fortschrittsfenster,

```
Ext.create(‚Shopware.window.Progress‘, {
    title: ‚Import wird ausgeführt ...‘,
    configure: function() {
        return {
            tasks: [{
                --> da wir mehrere Methoden ausführen, werden diese als Task konfiguriert
                    und für dem Eventlistener übergeben
                event: ‚import-products-process‘,
                        ->  die nach den Nomenklatur „[name-der-aktion]-process“  gebildeten
                            Prozessnamen  dienen als Alias für das jew. Event
                 data: Ext.clone(selection),
                text: ‚Produkt [0] von [1]‘
            }, {
                event: ‚change-create-date-process‘,
                data: Ext.clone(selection),
                text: ‚Setze Importdatum [0] von [1]‘
            },{
                event: ‚import-picture-process‘,
                data: Ext.clone(selection),
                text: ‚Importiere Artikelbild [0] von [1]‘
            }],

            infoText: ‚<h2>EBooks importieren</h2>‘ +
                ‚Je nach Datenmenge kann dieser Prozess einige Minuten dauern.‘
        }
    }
}).show();
}
});
```

Backend Plugin

Das Plugin kann wahlweise lokal (bitte direkt den Verzeichnisbaum kopieren) oder auf unter folgender URL getestet werden:

http://shopware.entwicklungsserver.de/backend/

User: dozent

Passwort: hs_offenburg_dozent

14. Zusammenfassung und Fazit

Diese Arbeit hat ein grosses Spektrum abgedeckt - betriebswirtschaftliche, bedarfsorientierte, konzeptionelle sowie technische Aspekte wurden untersucht. Ich habe klassische Software-Entwurfsmuster studiert sowie die Konzepte moderner MVC Frameworks untersucht und angewendet.
Dabei habe ich sowohl verschiedene Shopsysteme untersucht und bewertet. Ich habe die Systemarchitektur von Shopware 4 betrachtet wie auch die Konzepte der Frameworks, auf denen sie beruhen, analysiert und am Ende in der Pluginprogrammierung praktisch umgesetzt.
Dies war gleichzeitig spannend wie lehrreich, wenn auch der Einstieg in Symfony, Doctrine und Ext JS über den Weg „Shopware“ rückblickend nicht optimal gewählt war.
Um in diesem System frei abstrahieren zu können, kommt man ohne solides Vorwissen nicht aus. Nach Abschluss der Arbeit freue ich mich darauf, mir dieses an kleinen, selbstentwickelten Beispielanwendungen anzueignen.

15. Literaturliste

Buch (Monographie)

[1] *Open Source Jahrbuch 2008: Zwischen freier Software und Gesellschaftsmo dell, 1st ed. Berlin: Lehmanns, 2008.*

[2] *R. Allen, N. Lo, and S. Brown, Zend framework in action. Greenwich [Conn.]: Manning, 2009.*

[3] *Al-Qirim, Nabeel A. Y, Electronic commerce in small to medium-sized enter prises: Frameworks, issues and implications. Hershey, Pa: Idea, 2004.*

[4] *S. Angeli and W. Kundler, Der Online Shop - Handbuch für Existenzgründer Der Online-Shop: Handbuch für Existenzgründer ; Business-Plan, eShop-Systeme, ePayment, Behörden, Online-Recht, Marketing uvm. 4th ed. Mün chen/Germany: Markt & Technik; Markt und Technik, 2011 // 2006.*

[5] *H. Bidgoli, Electronic commerce: Principles and practice. San Diego: Academic Press, 2002.*

[6] *M. Dahm, Grundlagen der Mensch-Computer-Interaktion. München [u.a.]: Pearson Studium, 2006.*

[7] *R. Eggert, Zend Framework 2: Das Praxisbuch ; [Grundlagen, Komponenten, Module eigene Anwendungsmodule entwickeln ; für Umsteiger von Zend Framework 1 und Neueinsteiger ; zahlreiche Praxisbeispiele], 1st ed.: Galileo Press, 2013.*

[8] *R. Ekker, E-Commerce im stationaren Einzelhandel: Online und offline han del erfolgreich verbinden. [S.l.]: Bachelor + Master Publish, 2013.*

[9] *E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software: Pearson Education, 1994.*

[10] *E. Gamma, D. Riehle, R. Helm, R. Johnson, and J. Vlissides, Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software, 6th ed. Mün chen, Boston, Mass. [u.a.]: Addison-Wesley, 2011.*

[11] *J. Goll and M. Dausmann, Architektur- und Entwurfsmuster der Soft waretechnik: Mit lauffähigen Beispielen in Java: Springer Vieweg, 2014.*

[12] *G. Heinemann, Der neue Online-Handel: Erfolgsfaktoren und Best Practices, 4th ed. Wiesbaden: Gabler, 2012.*

[13] *J. Krisch and S. R. Rowold, E-Commerce für Fortgeschrittene: 50 Denkanstö ße für den Online-Handel von morgen, 2nd ed. Berlin: Epubli, 2011.*

[14] *A. Meier and H. Stormer, eBusiness & eCommerce: Management der digi talen Wertschöpfungskette, 3rd ed. Berlin, Heidelberg: Springer Berlin Hei delberg; Imprint: Springer, 2012.*

[15] *W. Sanders, Learning PHP Design Patterns: O'Reilly Media, 2013.*

[16] *E. Stahl, E-Commerce-Leitfaden: Noch erfolgreicher im elektronischen Han del, 3rd ed. Regensburg: Univ.-Verl. Regensburg, 2012.*

[17] *C. Strobel, Web-Technologien: in E-Commerce-Systemen // Web-Technologi en in E-commerce-Systemen. München, Wien: Oldenbourg, 2004.*

[18] *H. Tremp and M. Ruggiero, Application Engineering: Grundlagen für die objektorientierte Softwareentwicklung mit zahlreichen Beispielen, Aufga ben und Lösungen: Compendio Bildungsmedien, 2011.*

[19] *C. Valles, Zend Framework 2 Application Development. Birmingham: Packt Publishing, 2013.*

[20] *T. Walter, Kompendium der Web-Programmierung: Dynamische Web-Sites. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008.*

Internetdokument

[1] *AGOF, AGOF Branchenbericht Spezial „Bücher“ (2010). Available: http://www.agof.de/2010-spezial-bucher/.*

[2] *AIXPRO, Strategische und technische Vorteile von Shopware. Available: http://www.aixpro.de/strategische-und-technische-vorteile-von-shopware/.*

[3] *Alexander Graf, Das beste Shopsystem. Available: http://www.kassenzone.de/2012/03/25/das-beste-shopsystem/ (2014, Feb. 18).*

[4] *Ansible, YAML Syntax. Available: http://docs.ansible.com/YAMLSyntax.html (2014, Aug. 25).*

[5] *Commundus GmbH, Das wichtigste zu Open Source. Available: http://www.comundus.com/Loesungen/open-source/ (2014, Feb. 18).*

[6] *Doctrine API, Class ArrayCollection. Available: http://www.doctrine-project.org/api/common/2.3/class-Doctrine.Common.Collections.ArrayCollection.html (2014, Aug. 27).*

[7] *Fabien Potencier, Create your own framework… on top of the Symfony2 Components. Available: http://fabien.potencier.org/article/50/create-your-own-framework-on-top-of-the-symfony2-components-part-1 (2014, Aug. 13).*

[8] *Fabien Potencier, What is Symfony 2 ? Available: http://fabien.potencier.org/article/49/what-is-symfony2.*

[9] *Fabien Potencier, What is Dependency Injection ? Available: http://fabien.potencier.org/article/11/what-is-dependency-injection (2014, Jul. 23).*

[10] *Florian Schneider, Die 10 wichtigsten Usability-Regeln im E-Commerce. Available: http://www.netz98-blog.de/2010/06/21/die-10-wichtigsten-usability-regeln-im-e-commerce/ (2014, Apr. 08).*

[11] *Gablers Wirtschaftslexikon, Definition von Open Source. Available: http://wirtschaftslexikon.gabler.de/Definition/open-source.html (2014, Feb. 18).*

[12] *O. Groß, „Authentizität ist von zentraler Bedeutung“: Interview mit alphabetario.de. Available: http://www.shopbetreiber-blog.de/2013/02/20/authentizitaet-ist-von-zentraler-bedetung-interview-mit-alphabetario-de/ (2014, Feb. 17).*

[13] *http://docs.doctrine-project.org/, Doctrine 2 Dokumentation.*

[14] *ITWissen.info / Lexikon, Aspektorientierte Programmierung. Available: http://www.itwissen.info/definition/lexikon/Aspektorientierte-Programmierung-aspect-oriented-programming-AOP.html (2014, Aug. 23).*

[15] *K. Karimi, Entwicklung eines Evaluierungsmodells für E-Shop Software. Available: http://www.slideshare.net/oneduphine/388-3803992 (2014, Feb. 17).*

[16] *Mathew Setter, Zend Framework 2 Event Manager – A Gentle Introduction. Available: http://www.masterzendframework.com/tutorial/zend-framework-2-event-manager-a-gentle-introduction (2014, Aug. 23).*

[17] *Mathew Setter, Zend Framework 2 Modules – The Application’s Heart. Available: http://www.masterzendframework.com/tutorial/zend-framework-2-modules-the-applications-heart (2014, Aug. 23).*

[18] *Mathew Setter, Zend Framework 2 ServiceManager – Web Application Development Simplified. Available: http://www.masterzendframework.com/tutorial/zend-framework-2-servicemanager (2014, Aug. 23).*

[19] *Official Zend Documentation, Database and models. Available: http://framework.zend.com/manual/2.3/en/user-guide/database-and-models.html*

(2014, Aug. 23).

[20] Official Zend Documentation, Routing and controllers. Available: http://framework.zend.com/manual/2.3/en/user-guide/routing-and-controllers.html (2014, Aug. 23).

[21] Official Zend Documentation, The EventManager: The EventManager. Available: http://framework.zend.com/manual/2.0/en/modules/zend.event-manager.event-manager.html (2014, Aug. 23).

[22] Official Zend Documentation, Programmer's Reference Guide of Zend Framework 2. Available: http://framework.zend.com/manual/2.0/en/index.html (2014, Jul. 12).

[23] Official Zend Documentation, Introduction to the Module System. Available: http://framework.zend.com/manual/2.0/en/modules/zend.module-manager.intro.html (2014, Aug. 23).

[24] Official Zend Documentation, Modules. Available: http://framework.zend.com/manual/2.3/en/user-guide/modules.html.

[25] Sencha, Ext JS - Get up and running: Überblick über Ext JS's technische Dokumentation. Available: http://www.sencha.com/products/extjs/up-and-running (2014, Aug. 26).

[26] Sencha, Ext JS Examples. Available: http://dev.sencha.com/ext/5.0.1/examples/index.html.

[27] Sencha, Sencha Products. Available: http://docs.sencha.com/ (2014, Aug.26).

[28] Sencha, Try Sencha - Ext JS 4.2.0: Interaktive Beispiele für die Anwendung von Ext JS. Available: http://try.sencha.com/extjs/4.2.0/ (2014, Aug. 26).

[29] Sencha, Ext.grid.plugin.CellEditing Example. Available: http://try.sencha.com/extjs/4.2.0/docs/Ext.grid.plugin.CellEditing.1/viewer.html.

[30] Shopware AG, Plugin-Entwicklung: Boilerplate-Code. Available: http://wiki.shopware.de/Plugin-Entwicklung-Boilerplate-Code_detail_1077_869.html.

[31] Shopware AG, Shopware 4 - Grundlagen der Plugin Entwicklung. Available: http://wiki.shopware.de/_detail_971_867.html (2014, Aug. 21).

[32] Shopware AG, Shopware 4 Models. Available: http://wiki.shopware.de/Shopware-4-Models_detail_982_866.html#Insert_Event (2014, Aug. 27).

[33] Shopware AG, Shopware Controller. Available: http://wiki.shopware.de/Shopware-4-Controller_detail_970_866.html.

[34] Shopware AG, Shopware Developer`s Guide. Available: http://wiki.shopware.de/Developers-Guide_cat_796.html (2014, Aug. 27).

[35] Shopware AG, Shopware Store - Plugin „Entwicklertoolbar". Available: http://store.shopware.de/administration/entwicklertoolbar (2014, Aug. 27).

[36] Shopware AG, Suche. Available: http://wiki.shopware.de/_detail_929.html#Reiter_Relevanz.2FFelder.

[37] Stefan Hamann, Enlight - Shopware. Available: http://www.shopware.de/files/downloads/enlight-15652355.pdf (2014, Apr. 16).

[38] Symfony / SensioLabs, Controller. Symfony Controller. Available: http://symfony.com/doc/current/book/controller.html (2014, Aug. 25).

[39] Symfony / SensioLabs, Routing: Symfony Routing. Available: http://symfony.com/doc/current/book/routing.html (2014, Aug. 25).

[40] Symfony / SensioLabs, Symfony2 and HTTP Fundamentals. Available: http://symfony.com/doc/current/book/http_fundamentals.html (2014, Aug. 25).

[41] *Symfony / SensioLabs, The HttpFoundation Component (Details). Available: http://symfony.com/doc/current/components/http_foundation/introduction.html.*

[42] *Symfony / SensioLabs, Validation Constraints Reference. Available: http://symfony.com/doc/current/reference/constraints.html.*

[43] *Symfony / SensioLabs, HttpFoundation component: Symfony HttpFoundation component. Available: http://symfony.com/doc/current/components/http_foundation/index.html.*

[44] *Tim O'Reilly, Architecture of participation. Available: http://oreilly.com/pub/a/oreilly/tim/articles/architecture_of_participation.html (2014, Jul. 19).*

[45] *www.symfony.com, Symfony 2 cookbook. Available: http://symfony.com/doc/current/cookbook/index.html.*

[46] *Symfony / SensioLabs, The Symfony Book. Available: http://symfony.com/doc/current/book/index.html.*

[47] *Zend Documentation, Zend_Controller Grundlagen. Available: http://framework.zend.com/manual/1.12/de/zend.controller.basics.html (2014, Aug. 12).*

[48] *Zend Documentation, Zend Service Manager Quick Start Available: http://framework.zend.com/manual/2.0/en/modules/zend.service-manager.quick-start.html (2014, Aug. 23).*

[49] *AGOF, AGOF e. V. internet facts 2013-05 - Q3-2013_AGOF_facts figures Buecher.pdf. Available: http://www.agof.de/download-facts-figures-branchen berichte/.*

[50] *Gablers Wirtschaftslexikon, Electronic Shop. Available: http://wirtschaftslexikon.gabler.de/Definition/electronic-shop.html*

Pressemitteilung

[1] *Stefan Hamann / shoptrainer.de, Shopware 4 ab August mit vielen neuen Features und bis zu acht mal schneller. 2012.*

Persönliches Interview

[1] *Persönliches Interview mit Dr. Henne, Juli 2013*

16. Abbildungsverzeichnis

[16] *Sunrise alter Shop - Kategoriestruktur Auszug shop_alt_kategorie_01.png https://alt.sunrise-versand.de/bh/Shop/produkte_2010.cfm?page_id=1049&action=show_prodlist&cat1_id=35&cat2_id=1*

[17] *Sunrise alter Shop - Kategoriestruktur Übersicht shop_alt_kategorie_uebersicht.png https://alt.sunrise-versand.de/bh/Shop/produkte_2010.cfm?page_id=1049&action=show_prodlist&cat1_id=35&cat2_id=1*

[18] *Sunrise alter Shop - Kategoriestruktur Auszug Sunrise alter Shop - Kategoriestruktur Auszug https://alt.sunrise-versand.de/bh/Shop/produkte_2010.cfm?page_id=1049&action=show_prodlist&cat1_id=35&cat2_id=1*

[19] *Sunrise neuer Shop - Konzeption der Produktfilter - shop_neu_produktfilter.png Eigene Projektunterlagen*

[20] *Wichtigkeit der Suchfunktion in einem Online-Shop statista_wie_wichtig_ist_die_suche.eps http://de.statista.com/statistik/daten/studie/219229/umfrage/bedeutung-der-suchfunktion-in-online-shops-fuer-die-haendler/*

[21] *Konzeption der Produktfilter shop_neu_produktfilter.png Eigene Projektunterlagen*

[22] *Artikelsortiment-Migrationsmatrix shop_kategorie_produktfilter_matrix_auszug.eps - Eigene Projektunterlagen*

[23] *Migrations-Mapping (Kategorien) shop_neu_kategorie_mapping.eps Eigene Projektunterlagen*

[24] *Migrations-Mapping (Produkteigenschaften / Produktfilter)shop_neu_produktfilter_mapping.eps Eigene Projektunterlagen*

[25] *SQL Datenmigration sql_data_migration.eps Eigene Projektunterlagen*

[26] *Datenbanktabellen für Migration - shop_neu_DB_mapping.eps Eigene Projektunterlagen*

[27] *Aqua Data Studio - IDE für Datenbankoperationen IDEs_Aqua_Data_Studio.eps Eigene Projektunterlagen*

[28] *Shopware Youtube Webinare shopware_youtube_webinare.eps http://www.youtube.com/results?search_query=shopware+4+*

[29] *Doctrine Entity Queries - findOneBy doctrine_entity_manager_query_find_one_by.eps http://docs.doctrine-project.org/en/2.0.x/reference/working-with-objects.html*

[30] *Deutschland - Umsätze mit Buechern im Versandhandel (2012). statista_umsatze_mit_buechern_versandhandel_2012.eps http://statista-research.com/publikationen/e-commerce-markt-deutschland-2013/*

[31] *PHP Storm 7 Icon PhpStorm7.png http://blog.jetbrains.com/phpstorm/files/2013/10/PhpStorm7_blog_intro.png*

[32] *PHP Storm 7 Icon alternativ PhpStorm7_alternativ.png http://confluence.jetbrains.com/display/PhpStorm/Getting+Started+with+PhpStorm+as+Google+App+Engine+PHP+IDE*

[33] *Dependency Injection - Klasse „SessionStorage“ zend_dependency_injection_01.EPS http://fabien.potencier.org/article/11/what-is-dependency-injection*

[34] *Dependency Injection - Klasse „User“ zend_dependency_injection_02.EPS http://fabien.potencier.org/article/11/what-is-dependency-injection*

[35] *SessionStorage innerhalb Klasse User - zend_dependency_injection_02b.EPS*

http://fabien.potencier.org/article/11/what-is-dependency-injection

[36] Dependency Injection - SessionStorage mit Dependency Injection depen dency_injection_03.EPS http://fabien.potencier.org/article/11/what-is-dependency-injection

[37] Dependency Injection - Übersicht zend_dependency_injection_04.EPS http://fabien.potencier.org/article/11/what-is-dependency-injection

[38] Dependency Injection - Übersicht zend_dependency_injection_05.EPS http://fabien.potencier.org/article/11/what-is-dependency-injection

[39] Zend Modul Code Beispiel zend_module_code_example.eps http:// framework.zend.com/manual/2.3/en/user-guide/modules.html

[40] GoF Entwurfsmuster - Proxie - Beispiel „Bild“entwurfmuster_gof_proxie_ex ample.eps Erich Gamma, Richard Helm, Ralph Johnson und John Vlissi des - Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software

[41] GoF Entwurfsmuster - Observer - Beispiel „Zeitung“ design_patterns_obser ver_example.png http://www.philipphauer.de/study/se/design-pat tern/observer.php

[42] Symfony - Grundlagen - Request symfony_basic_concepts_request.eps http://symfony.com/doc/current/book/http_fundamentals.html

[43] Symfony - Grundlagen - Response symfony_basic_concepts_response. eps http://symfony.com/doc/current/book/http_fundamentals.html

[44] The Symfony Application Flow symfony-request-flow.png http:// symfony.com/doc/current/book/http_fundamentals.html

[45] Symfony - Routing - YAML symfony_routing_code_example_yaml.eps http://symfony.com/doc/current/book/http_fundamentals.html

[46] Symfony - Routing - XML symfony_routing_code_example_xml.eps http://symfony.com/doc/current/book/http_fundamentals.html

[47] Symfony - Routing - PHP symfony_routing_code_example_php.eps http://symfony.com/doc/current/book/http_fundamentals.html

[48] Symfony - Routing - Varianten symfony_routing_code_example_sing le_entry.eps http://symfony.com/doc/current/book/from_flat_php_to_ symfony2.html

[49] Doctrine Codebeispiel Beziehungen doctrine_code_example_02.eps http://marco-pivetta.com/doctrine2-orm-tutorial/

[50] Doctrine Codebeispiel Tabellendefinition doctrine_code_example_03. eps http://marco-pivetta.com/doctrine2-orm-tutorial/

[51] Doctrine - Definition Model - Auszug doctrine_define_model_01.eps Eigene Projektunterlagen

[52] Doctrine - Codebeispiel Pagination doctrine_code_example_pagination. eps http://doctrine-orm.readthedocs.org/en/latest/tutorials/paginati on.html

[53] Shopware - Backend - Übersicht 01 shopware_backend_overview_01.eps Eigene Projektunterlagen

[54] Shopware - Backend - Übersicht 02 shopware_backend_overview_02.eps Eigene Projektunterlagen

[55] Shopware - Backend - Schnellsuche shopware_backend_search_article_ customer.eps Eigene Projektunterlagen

[56] Shopware - Backend - Grundeinstellungen shopware_backend_basic_op tions.eps Eigene Projektunterlagen

[57] Shopware - Backend - Media Manager shopware_backend_media_manager.eps Eigene Projektunterlagen

[58] Shopware - intelligente Suche 01 shopware_backend_intelligent_search_01.eps Eigene Projektunterlagen

[59] Shopware - intelligente Suche 02 shopware_backend_intelligent_search_02.epsEigene Projektunterlagen

[60] Smarty - Syntax Beispiel smarty_syntax_01.eps Eigene Projektunterlagen

[61] Shopware - Textbausteine shopware_backend_textbausteine.eps Eigene Projektunterlagen

[62] Shopvare - Sprachvariable smarty_sprachvariable.eps Eigene Projekt unterlagen

[63] Ext JS - Try - Beispiel „Cell Editing“ extJS_try.eps http://try.sencha.com/extjs/4.2.0/docs/Ext.grid.plugin.CellEditing.1/viewer.html

[64] Shopware -. Plugin Entwicklertoolbarshopware_entwicklertoolbar.eps Eigene Projektunterlagen

[65] Shopware -. Debug mit Fire-PHP shopware_debug_with_fire_php.eps Eigene Projektunterlagen

[66] Frontend - Layout - Mockup „Startseite“ mockup_sunrise_shop.ai Eigene Projektunterlagen

[67] Frondend - grafische Kategorieansicht shopware_frontend_category_overview.eps Eigene Projektunterlagen

[68] Shopware Einkaufwelten shopware_einkaufswelten.eps Eigene Projektunterlagen

[69] Frontend - Hauptkategorie Produktempfehlungen shopware_frontend_single_category.eps Eigene Projektunterlagen

[70] Frontend - Use Case Materia Medica 01 frontend_materia_medica_01.eps Eigene Projektunterlagen

[71] Frontend - Use Case Materia Medica 02 frontend_materia_medica_02.eps Eigene Projektunterlagen

[72] Shopware Ordnerstruktur shopware_ordnerstruktur.epshttp://wiki.shopware.de/Shopware-4-Ordnerstruktur_detail_969_866.html

[73] Shopware ER Diagramm Shopware_ER_Diagramm.pdf http://wiki.shopware.de/Shopware-4-ER-Diagramm_detail_1331_869.html

[74] Plugin - Dateistruktur plugin_overview_files.eps Eigene Projektunterlagen

[75] Frontend Suche 01 frontend_suche_00.eps Eigene Projektunterlagen

[76] Frontend Suche 02 frontend_suche_01.eps Eigene Projektunterlagen

[77] Frontend Suche 03 frontend_suche_02.eps Eigene Projektunterlagen

[78] Frontend Buttons frontend_button_types.eps Eigene Projektunterlagen

[79] Plugin 01 plugin_step_01.eps Eigene Projektunterlagen

[80] Plugin 02 plugin_step_02.eps Eigene Projektunterlagen

[81] Plugin 03 plugin_step_03.eps Eigene Projektunterlagen

[82] Plugin 04 plugin_step_04.eps Eigene Projektunterlagen

Abbildung „Evaluierung ERP Systeme“ [86]

Evaluierung - ERP Systeme (Stand Juni 2013, Einstufung der Leistungen anhand Hersteller-Demoversionen / Dokumentation)

Name	Kurzbeschreibung	Technologie	URL	DATEV-Schnitstelle	Lizenzmodell	Dokumentation	Schnittstellen zu E-Shops	Versand (z.B. Paket-schein-erstellung inkl. Barcode)
Evolution	Gesamtlösung für alle Bereiche des Unternehmens Version eBusiness:Komponenten ERP-Lösung (Warenwirtschaft, Lagerverwaltung, FIBU), Webshop und Content Management System	.NET	http://www.eevolution-erp.de	x	verschiedene Modelle je nach Bedarf	***	bietet integrierten Shop	UPS, Trans-o-flex, DHL, DPD, System-Gut
Nuclos	Modulares ERP-Baukastensystem	Java	http://www.nuclos.de	x	kostenlos ohne Support je nach Bedarf	***	Magento OXID xt:commerce	nachrüstbar über Erweiterung
Fakturama	einfach gehaltenes Waren-wirtschaftssystem auf OpenSource Basis	Java	http://fakturama.sebulli.com	x	kostenlos	*	osCommerce xt:Commerce xtcModified	Übertragung Adressdaten an Paketdienste
PIXI	ERP mit Spezialisierung auf E-Commerce	PHP	http://www.pixi.eu	x (ab Professionell Version)	wahlweise als Mietlösung bei PIXI oder Professional Version mit eigenem Server (ab 10.000 Euro)	***	Magento OXID Shopware XT Commerce und weitere	alle gängigen Anbieter
osRetail	Open Source ERP mit Versandhandels-ausrichtung	Java	http://osretail.de	per Einbindung vonGnuCash und Erweiterungs-modulen (Open Source)	Installation oderSaaS/Cloud LösungPreise auf Anfrage / je nach Umsatz (eher günstig)	***	Magento Konakart	keine Informationen

	A	B	C	D	E
1	Hauptkategorie alt	Unterkategorie alt	Hauptkategorie neu	Unterkategorie neu	Produktfilter
2	Displays und Sortiersysteme!		Praxiszubehör	Weiteres Zubehör	Non-Books: Sortiersysteme
3	Einsteiger und Laien		Bücher	Einführung	Niveau: Einsteiger / Laien
4	Englische Literatur		Bücher	Englisch	
5	Hahnemann		Quellen	S. Hahnemann	Theorie: Quellen
6	Homöopathie-Produkte	Büsten, Miniaturen	Praxiszubehör	Weiteres Zubehör	Non-Books: Büsten / Poster
7		Bucheinbände	Praxiszubehör	Weiteres Zubehör	Non-Books: Rund ums Buch
8		Ledertaschen	Praxiszubehör	Lederetuis	Non-Books: Lederetuis
9		Displays und Sortiersysteme	Praxiszubehör	Weiteres Zubehör	Non-Books: Sortiersysteme
10		Mörser	Praxiszubehör	Weiteres Zubehör	Non-Books: Mörser
11	Kassetten, CDs, Videos	Videos, DVDs	Software / Multimedia	Video-DVDs	Non-Books: Multimedia
12		Freiburger Homöopathie Tage	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
13		Boller Vorträge	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
14		Andreas Krüger	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
15		Willibald Gawlik	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
16		Sonstige Kassetten, CDs	Software / Multimedia	Audio-CDs	Non-Books: Multimedia
17	Materia Medica	Enzyklopädien	Bücher	Materia Medica	Niveau: Therapeuten
18		Arzneimittellehren	Bücher	Materia Medica	
19		Leitsymptome	Bücher	Materia Medica	Theorie: Leitsymptome
20		Arzneimittelbeschreibungen	Bücher	Materia Medica	
21		Einzelne Arzneimittel	Bücher	Materia Medica	Theorie: Einzelne Arzneimittel
22		ältere Werke	Quellen	Weitere Quellen	Theorie: Quellen
23	Medizin	Heilpraktiker / Ausbildung	Bücher	Medizin	Theorie: Ausbildung (HP) / Weitere Ther
24		Heilpraktiker / Prüfung	Bücher	Medizin	Theorie: Prüfung (HP) / Weitere Therapi
25		Weiterführende Literatur	Bücher	Medizin	Weitere Therapien: Medizin / Weitere Th
26	Patientenkarteimappen / Fragebögen				
27	Naturheilkunde	Phytotherapie	Bücher	Weitere Therapien	Weitere Therapien: Phytotherapie
28		Bachblüten	Bücher	Weitere Therapien	Weitere Therapien: Bach-Blüten
29		Akupunktur, TCM	Bücher	Weitere Therapien	Weitere Therapien: TCM / Akupunktur
30		Biochemie nach Dr. Schüßler	Bücher	Weitere Therapien	Weitere Therapien: Biochemie (Schüßle
31		Ganzheitliche Tiermedizin	Bücher	Weitere Therapien	Praxis: Tiere / Pflanzen
32		Allgemeines, Leitfäden	Bücher	Weitere Therapien	Weitere Therapien: Heilpraktiker
33		Antlitzdiagnostik	Bücher	Weitere Therapien	Theorie: Anamnese / Weitere Therapien:
34		Ernährung	Bücher	Weitere Therapien	Weitere Therapien: Ernährung
35		Sonstiges	Bücher	Weitere Therapien	
36					
37	Neuerscheinungen		Bücher	Neuerscheinungen	Sonderpreise: Neuerscheinungen
38	Patientenkarteimappen / Fragebögen		Praxiszubehör	Patientenkarteimappen	Non-Books: Patientenkarteimappen
39	Praxis	Therapeutische Leitfäden	Bücher	Praxis	Praxis: Therapie (Allgemein)
40		Kasuistik	Bücher	Praxis	Praxis: Fallsammlungen
41		Frauen	Bücher	Praxis	Praxis: Frauen
42		Kinder	Bücher	Praxis	**Praxis: Kinder**
43		Tierhomöopathie	Bücher	Praxis	Praxis: Tiere / Pflanzen
44		Homöopathie für die Seele	Bücher	Psychologie	Praxis: Psychologie
45		Krebs	Bücher	Praxis	Praxis: Onkologie
46		Sonstige Fachgebiete	Bücher	Praxis	Praxis: Therapie (Kopf -> Fuß)
47	RADAR Preisaktion		Software / Multimedia	Radar	Non-Books: Software
48	RadarOpus Homöopathie-Software	Programm-Pakete	Software / Multimedia	Radar	Non-Books: Software
49		Easy-Programme	Software / Multimedia	Radar	Non-Books: Software
50		Module	Software / Multimedia	Radar	Non-Books: Software
51		Repertorien	Software / Multimedia	Radar	Non-Books: Software
52		Updates	Software / Multimedia	Radar	Non-Books: Software
53		Upgrades	Software / Multimedia	Radar	Non-Books: Software
54		Bibliothek Opus	Software / Multimedia	Radar	Non-Books: Software
55		Zusatz-Lizenzen	Software / Multimedia	Radar	Non-Books: Software
56		Service / Schulung	Software / Multimedia	Radar	Non-Books: Software
57	Repertorien	Repertorien	Bücher	Repertorien	
58		Arzneimittelbeziehungen	Bücher	Materia Medica	Theorie: Handbücher (AML)
59		Wörterbücher, Indizes	Bücher	Repertorien	
60					
61	Sonderpreise	Preisaktionen	Bücher	Sonderpreise	Sonderpreise: Preisaktionen
62		Paketpreise	Bücher	Sonderpreise	Sonderpreise: Preisaktionen
63		Modernes Antiquariat	Bücher	Sonderpreise	Sonderpreise: Antiquariat
64		Preishits	Bücher	Sonderpreise	Sonderpreise: Preisaktionen
65		Mängelexemplare	Bücher	Sonderpreise	Sonderpreise: Antiquariat
66	Spitzentitel des Jahres (fällt als Kategorie weg, weil im	n.a.	Bücher	manuell zuordnen	
67	Sunrise_Buecherregal	für den Start	Bücher	manuell zuordnen	Sunrise-Bücherregal: Einsteiger / Laien
68		für Studium und Ausbildung	Bücher	manuell zuordnen	Sunrise-Bücherregal: Studium / Ausbild
69		für Praxis und Weiterbildung	Bücher	manuell zuordnen	Sunrise-Bücherregal: Therapeuten
70					
71	Taschenapotheken	Lederetuis konventionell	Praxiszubehör	Lederetuis	Non-Books: Lederetuis
72	(Kategorie wird durch Praxiszubehör ersetzt)	Sunrise-Lederetuis	Praxiszubehör	Sunrise-Lederetuis	Non-Books: Lederetuis / Non-Books: Su
73		Sunrise-Stoffrollen	Praxiszubehör	Sunrise-Stoffrollen	Non-Books: Sunrise Produkte
74		Homöo-Set Lederetuis	Praxiszubehör	Homö-Set Lederetuis	Non-Books: Lederetuis
75		Nappa-Lederetuis	Praxiszubehör	Lederetuis	**Non-Books: Lederetuis**
76		Weitere Etuis	Praxiszubehör	Weitere Etuis	

Abbildung „Artikelsortiment-Migrationsmatrix“ [22]